Volume 10 Number 4

ACTA CYBERNETICA

Szeged, 1992

Information for authors: Acta Cybernetica publishes only original papers in English in the field of computer sciences. Review papers are accepted only exceptionally. Manuscripts should be sent in triplicate to one of the Editors. The manuscript must be typed double-spaced on one side of the paper only. For the form of references, see one of the articles previously published in the journal.

Acta Cybernetica, Vol. 10, Nr. 4, Szeged, 1992

A criterion for the simplicity of finite Moore automata

A. Ádám*†

Abstract

A Moore automaton $\mathbf{A} = (A, X, Y, \delta, \lambda)$ can be obtained in two steps: first we consider the triplet (A, X, δ) – called a semiautomaton and denoted by $\mathbf{S}$ – and then we add the components Y and λ which concern the output functioning. Our approach is: $\mathbf{S}$ is supposed to be fixed, we vary λ in any possible way, and – among the resulting automata – we want to separate the simple and the nonsimple ones from each other. This task is treated by combinatorial methods. Concerning the efficiency of the procedure, we note that it uses a semiautomaton having $|A|(|A|+1)/2$ states.

1 Introduction and terminology

§ 1.

The question, when a Moore automaton is simple, has already been the subject of a series of previous papers.[1] Let some earlier results be outlined.[2] If, particularly, only autonomous automata are considered (i.e., $|X| = 1$ is required), the question has been solved in [4] as a consequence of the theory describing all congruences of autonomous automata. Without the restriction to autonomousness, a result of certain theoretical importance has been obtained in [2]; this statement does not seem to be worthy practically, because its algorithmic complexity depends on $|A|$ exponentially. Investigations of recursive character are contained in [5] and [6], the general problem of simplicity was there reduced to the question, when a strongly connected automaton (i.e., an automaton having no proper subautomata) is simple.

In the present considerations the problem of simplicity is dealt with for the eintirety of automata, we rely on the result achieved in [2]. We choose the way that first a semiautomaton $\mathbf{S} = (A, X, \delta)$ is thought to be fixed, and we form

*MTA Matematikai Kutatóintézet, H-1364 Budapest, P.O.Box 127. Hungary

†Research partially supported by the Hungarian National Foundation for Scientific Research (OTKA) grant, no. 1909.

[1]The researcher of this problem feels his situation to be similar to that of a mountain-climber who besieges a difficultly reachable peak from various sides, since he does not know in advance where he must turn back because of a too steep rise.

[2]Out of the three results to be mentioned now, the first and second ones are restated in this paper as Propositions D and A (in § 11 and § 3, respectively).

then several automata $\mathbb{A}_\lambda = (A, X, Y, \delta, \lambda)$ so that the output components Y, λ are prescribed in every (essentially different) manner. We obtain a necessary and sufficient condition that separates the simple $\mathbb{A}_\lambda'$s from the nonsimple ones. We use combinatorial tools, and our considerations are in connection with the articles [7], [8] where partial results were gained in common with I. Babcsányi and F. Wettl.

Sketching the content of this paper, let it be mentioned first that the terminology concerning automata is introduced in §§ 2, 3; together with restating some former results. A glance is thrown at the graph theory in § 4.

The construction, elaborated in § 5, and the Theorem, exposed at the end of § 6, are the principal purport of the article. § 7 contains the proof of the Theorem and of two cognate propositions.

The condition for the simplicity of automata, asserted in the Theorem, allows sometimes a useful further analysis by logical methods; an insight into this possibility is explained in § 8. In § 9 examples are treated on how the Theorem can be applied in practice.

The paper terminates with touching some questions on combinatorial complexity, arising if the method is applied. These considerations do not set up a claim for completeness at all, they are of intuitive nature. The possibilities of future contributions to this topics are specified as open problems in § 11. It is probable that a genuine expert of the combinatorial complexity theory may conceive essential further thoughts in addition to the ideas formulated in §§ 10, 11.

§ 2.

As usual, we understand by a *finite Moore automaton* a quintuple $\mathbb{A} = (A, X, Y, \delta, \lambda)$ where A, X, Y are finite sets (called the set of states, set of input symbols, set of output symbols, respectively), δ (the transition function) is a mapping of $A \times X$ into A and λ (the output function) is a mapping of A onto Y.

The finite sequences (of arbitrary nonnegative length) consisting of elements of X are called input words. The set of all input words is denoted by $F(X)$. The meaning of $\delta(a, p)$ is the customary where p is an input word.

Let a and b be two states of an automaton. We say that b is *accessible* from a if there exists an input word p such that $\delta(a, p) = b$. The accessibility is a reflexive and transitive relation. If any of a, b is accessible from the other, then it is said that a and b are *mutually accessible*. The mutual accessibility is an equivalence relation in A, the equivalence classes are called the *strongly connected blocks* – or, for the sake of brevity, the blocks – of A. If there is only one block, we say that A is a *strongly connected* automaton.

Let π be an arbitrary equivalence relation in A. π is called a *congruence* (of A) if $a \equiv b \pmod{\pi}$ implies the formulae $\delta(a, x) \equiv \delta(b, x) \pmod{\pi}$ and $\lambda(a) = \lambda(b)$ whenever $a \in A, b \in A, x \in X$. The minimal partition of A is the *trivial congurence* of $\mathbb{A}$. It is said that $\mathbb{A}$ is *simple* (or *reduced*) if $\mathbb{A}$ has no nontrivial congruence.

If we do not take into consideration the third component Y and the fifth component λ of a Moore automaton $\mathbb{A} = (A, X, Y, \delta, \lambda)$, then the resulting structure $\mathbf{S} = (A, X, \delta)$ is called a *semiautomaton*.[3] We say then that $\mathbf{S}$ is the *scheme* (or *projection*) of $\mathbb{A}$ and, reciprocally, that $\mathbb{A}$ is an *automaton completion* (or a-completion) of $\mathbf{S}$. Of course, a semiautomaton $\mathbf{S}$ has many a-completions, depending on how λ is chosen. We shall use the notation $\mathbb{A}_\lambda$ sometimes when the a-completion of a semiautomaton with the output function λ is regarded.

A pair (a, b) is called a *proper* pair if $a \neq b$.

[3] The present use of the word "semiautomaton" differs from the terminology of [10].

§ 3.

Throughout this section let (a, b) be a (proper or nonproper) unordered pair of states of a Moore automaton.

We denote by $H_{a,b}$ the set of all input words p satisfying $\delta(a,p) \neq \delta(b,p)$. It is said that

(a,b) is a pair of *first type* if $|H_{a,b}| < \infty$,
(a,b) is a pair of *second type* if $H_{a,b} = F(X)$,
(a,b) is a pair of *third type* if $H_{a,b} \subset F(X)$ and $|H_{a,b}| = \infty$.

The difference set $F(X) - H_{a,b}$ is obviously either empty of infinite. The pairs of second and third type are necessarily proper.

We say that (a,b) is a *distinguishable* pair if there exists an input word p such that $\lambda(\delta(a,p)) \neq \lambda(\delta(b,p))$. In the contrary case (a,b) is *indistinguishable*. The relation of indistinguishability, to be denoted by $\pi_{\max}$, is clearly an equivalence relation. The following fact establishes a connection between simplicity and distinguishability (see [2], § 5; [5], § 4):

Proposition A. *Consider $\pi_{\max}$ in a Moore automaton* **A**. *The relation $\pi_{\max}$ is a congruence of* **A** *and each congruence of* **A** *is a refinement of $\pi_{\max}$.* **A** *is simple if and only if $\pi_{\max}$ equals the minimal partition of A (or, equivalently, if each proper pair (a,b) is distinguishable where $a \in A, b \in A$).*

If a proper pair (a,b) of states is indistinguishable and of first type, then we say that a and b are *weakly indistinguishable*. If a pair (a,b) is indistinguishable and of second type, then we say that a and b are *strongly indistinguishable*. If (a,b) is indistinguishable and of third type, then we say that a and b are *compoundly indistinguishable*.

The three kinds of indistinguishability introduced above are pairwise excluding. The subsequent assertion follows from [8], Proposition 2:

Proposition B. *The weak indistinguishability is a transitive relation.*

The analogous statement does not hold for the other two indistinguishability types (cf. [8], Chapter III).

Let (a,b) be a state pair. If $\lambda(a) = \lambda(b)$ holds and $\delta(a,x) = \delta(b,x)$ is valid for every $x(\in X)$, then we say that (a,b) is an *associated* pair. The relation of being associated is an equivalence in A.

It is clear that any associated proper pair is weakly indistinguishable. The converse of this fact does not hold (in general), but we have the following sentence (see [7], Proposition 2):

Proposition C. *Consider the state pairs in a Moore automaton. There is a proper associated pair if and only if there is a weakly indistinguishable pair.*

§ 4.

It is not superfluous to say here a few words on graph theory, because we shall construct a nondirected graph at the end of § 5, and our automaton-theoretical

considerations use sometimes certain ideas that originate from the theory of directed graphs. Let [11], [13] be mentioned as reference books of the two main branches of graph theory.

The graph got in § 5 is simple in the sense that each vertex pair [or, in another terminology, point pair] is joined by at most one edge [line], and each edge joins two different vertices. We use the notation $[ab]$ for the edge joining a and b.

The notions of accessibility (introduced in § 2) correspond precisely to the analogous concepts in directed graph theory (for the latter, see e.g. Chapter 3 of [13]). One can show easily that we get a cycle-free directed graph if we form the condensation of the strongly connected blocks [strong components] in a directed graph ([13], Theorem 3.6). Keeping this fact in mind, the reader may perhaps understand better Steps 2-4 of the Construction of § 5.

2 Results

§ 5.

Let $\mathbf{S} = (A, X, \delta)$ be a semiautomaton where $|A| \geq 2$. $\mathbf{S}$ is regarded to be fixed in Chapter 2. We denote $|A|$ by v.

If the output function $\lambda : A \to Y$ is varied, we can get several automaton completions $\mathbf{A}_\lambda = (A, X, Y, \delta, \lambda)$ from $\mathbf{S}$. Our aim is to examine the question: when is a simple $\mathbf{A}_\lambda$ obtained (depending on the choice of λ). Among the automata $\mathbf{A}_\lambda$, it is yielded always a simple one (if $|Y| = v$ and λ is bijective), and also a nonsimple one (if $|Y| = 1$).

In the next construction, we are going to establish a pair (G, ρ) where G is a nondirected graph (whose vertex set equals A) and ρ is a partition of the edges of G.

CONSTRUCTION. The procedure consists of five steps.

Step 1. Let a semiautomaton $\mathbf{R} = (C, X, \delta_R)$ be introduced in the following manner: let C be the set of all (proper and nonproper) unordered pairs (a, b) where $a \in A, b \in A$, define δ_R by the rule

$$\delta_R((a, b), x) = (\delta(a, x), \delta(b, x)). \tag{1}$$

Comments to Step 1. The right-hand side of (1) is meant as an unordered pair. Clearly $|C| = v(v+1)/2$. If (a, b) is a nonproper pair, then the values $\delta_R((a, b), x)$ are again nonproper pairs, hence $\mathbf{R}$ has a subsemiautomaton isomorphic to $\mathbf{S}$. In the terminology of products of automata, we can say that the factor semiautomaton $(\mathbf{S} \otimes \mathbf{S})/\sigma$ is denoted by $\mathbf{R}$ where $\otimes$ is the sign of direct product and σ is the congruence of $\mathbf{S} \otimes \mathbf{S}$ defined by the rule that $(a, b) \equiv (c, d) \pmod{\sigma}$ exactly if either $a = c$, $b = d$ are true or $a = d, b = c$ hold.

Step 2. Denote by ε the equivalence relation of mutual accessibility in C.

Comment to Step 2. If K is an equivalence class modulo ε, then either each element of K is a proper pair or each element of K is a nonproper pair.

Step 3. Consider the equivalence classes K modulo ε (in C) satisfying the conditions (a) and (b):

(a) K consists of proper pairs,

(b) whenever $(a, b) \in K$ and $x \in X$, then

$$\delta_R((a, b), x) \in K. \tag{2}$$

Denote the number of these classes by j and themselves the classes by $K_1, K_2, \ldots, K_j$.

Step 4. Consider the equivalence classes K modulo ε (in C) such that K does not satisfy (b), it fulfils (a) and the following condition (c):

(c) whenever $(a, b) \in K$ and $x \in X$, then either $\delta_R((a, b), x)$ is a nonproper pair or (2) is true.

Denote the number of these classes by k and themselves the classes by $K_{j+1}, K_{j+2}, \ldots, K_{j+k}$.

Comments to Steps 3, 4. Condition (b) can be expressed by saying that K determines a subsemiautomaton of $\mathbf{R}$. The ordering of the classes $K_1, \ldots, K_j$ is arbitrary and the same holds for $K_{j+1}, \ldots, K_{j+k}$. The $j + k$ classes are pairwise disjoint because they have arisen as different classes of an equivalence relation. The number $j + k$ is positive by the finiteness of C.

Step 5. Denote by G the nondirected graph whose vertex set is A and in which two vertices a, b are joined by an edge $[ab]$ precisely when

$$(a, b) \in K_1 \cup K_2 \cup \ldots \cup K_{j+k}.$$

Moreover, let the edge $[a, b]$ belong to the i-th class (modulo ρ), L_i, exactly when $(a, b) \in K_i$ (where $1 \leq i \leq j + k$).

§ 6.

We state two propositions and a theorem on an arbitrary a-completion $\mathbf{A}_\lambda = (A, X, Y, \delta, \lambda)$ of $\mathbf{S}$ and on the partitioned graph (G, g). The verification of the results will be done in the next section.

Proposition 1 *The following two assertions are equivalent:*

(α) *There is a strongly indistinguishable state pair in* $\mathbf{A}_\lambda$.

(β) *There exists a number* i, *fulfilling* $1 \leq i \leq j$, *such that, whenever* $[ab] \in L_i$, *then* $\lambda(a) = \lambda(b)$.

Proposition 2 *If there is a weakly indistinguishable state pair in* $\mathbf{A}_\lambda$, *then there exists a number* i *such that the subsequent assertions are true:*

$$j + 1 \leq i \leq j + k,$$

$$|L_i| = 1, \text{ and}$$

we have $\lambda(a) = \lambda(b)$ *for the single element* $[ab]$ *of* L_i.

We have arrived to the main result of the paper.

Theorem 1 *Let an output function $\lambda : A \to Y$ be added to* **S**. *The following two conditions are equivalent for the resulting automaton* $\mathbf{A}_\lambda$*:*

(I) $\mathbf{A}_\lambda$ *is simple.*

(II) In any class L_i *(where* $1 \leq i \leq j+k$*) there exists at least one edge* $[a_i b_i]$ *such that* $\lambda(a_i) \neq \lambda(b_i)$.

§ 7.

Proof of Proposition 1. $(\alpha) \Rightarrow (\beta)$. To any (unordered) proper state pair (a, b) let us denote by $Q(a, b)$ the set of the (proper and nonproper) state pairs (c, d) which satisfy

$$(c, d) = (\delta(a, p), \delta(b, p))$$

with some $p(\in F(X))$. First we mention immediate consequences of this definition. We have $(a, b) \in Q(a, b)$. The pair (c, d) is accessible from (a, b) if and only if $Q(c, d) \subseteq Q(a, b)$.

$$(a, b) \equiv (c, d) (\mathrm{mod}\ \varepsilon)$$

if and only if $Q(a, b) = Q(c, d)$. If (a, b) is a strongly indistinguishable pair and $(c, d) \in Q(a, b)$, then also (c, d) is strongly indistinguishable.

Consider now a strongly indistinguishable state pair (a, b) in $\mathbf{A}_\lambda$. We can choose a pair (c_0, d_0), belonging to $Q(a, b)$, in such a manner that the strict inclusion

$$Q(c, d) \subset Q(c_0, d_0) \tag{3}$$

is false for any $(c, d)(\in Q(a, b))$. (This choice is possible by the finiteness of **R**.)

Denote $Q(c_0, d_0)$ by K. The condition (b) in § 5 is obviously valid for K and K consists of strongly indistinguishable pairs only, furthermore our condition on the falsity of (3) implies that K is just a complete class modulo ε. Consequently, K equals one of the classes $K_1, K_2, \ldots, K_j$ (introduced in Step 3 of the Construction), thus the validity of (β) is clear.

$(\beta) \Rightarrow (\alpha)$. Suppose (β) for a number i, consider an arbitrary edge $[ab]$ in L_i. We can see easily that $Q(a, b) = K_i$, hence (a, b) is a strongly indistinguishable pair.

Proof of Proposition 2. Assume the existence of a weakly indistinguishable pair. Then there is (by Proposition C) a proper associated state pair $(a, b).\lambda(a) = \lambda(b)$ is clear. The one-element set $\{(a, b)\}$ is evidently a class $K_i(\mathrm{mod}\ \varepsilon)$ and i fulfils $j < i \leq j + k$.

Proof of the Theorem.

First we show that the falsity of (I) implies the falsity of (II). Denote the set of indistingushable state pairs of $\mathbf{A}_\lambda$ by J. If $\mathbf{A}_\lambda$ is not simple, then $\pi_{\max}$ differs from the minimal partition of A (by Proposition A in § 3), therefore $J \neq \emptyset$. We separate three cases (the first and second ones can overlap each other).

Case 1: J contains a strongly indistinguishable pair. Proposition 1 applies, the truth of (β) shows that (II) does not hold.

Case 2: J contains a weakly indistinguishable pair. We get now by Proposition 2 that (II) is not fulfilled.

Case 3: any element of J is compoundly indistinguishable. Recall the notation $Q(a, b)$ (where $(a, b) \in J$). Define $Q'(a, b)$ as the difference set $Q(a, b) - P$ where P is the set of nonproper state pairs. We have always $(a, b) \in Q'(a, b) \subseteq$

J. Analogously to the first proof in § 7, we start with an arbitrary $(a, b)(\in J)$ and we choose a $(c_0, d_0)(\in J)$ such that

$$Q'(c, d) \subset Q'(c_0, d_0)$$

is false when (c, d) is an arbitrary element of $Q'(c_0, d_0)$. It is obtained that $Q'(c_0, d_0)$ is one of the classes $K_1, K_2, \ldots, K_{j+k}$, say, K_i. (II) is not satisfied with this i because $\lambda(a_i) = \lambda(b_i)$ whenever $(a_i, b_i) \in Q'(c_0, d_0)$.

Conversely, let us assume that (II) is not fulfilled. There is an i such that $[ab] \in L_i$ (that is, $(a, b) \in K_i$) implies $\lambda(a) = \lambda(b)$. Remember how K_i has been constructed in § 5. Choose an arbitrary element (a_0, b_0) of K_i. Whenever $(c, d) \in Q(a_0, b_0)$, then either $(c, d) \in K_i$ or $c = d$; we get $\lambda(c) = \lambda(d)$ in both cases. We have shown that (a_0, b_0) is an indistinguishable proper pair. Thus $\pi_{\max}$ is not the minimal partition of A, hence (by Proposition A in § 3) $\mathbf{A}_\lambda$ is not simple.

3 Discussion and examples

§ 8.

Suppose that we consider some semiautomaton $\mathbf{S}$ and we want to use the Theorem for getting an overview of the simple automata among all the automata obtained as $\mathbf{A}_\lambda$.

There is no difficulty if the graph G and its edge-partition ρ are enough perspicuous. In the contrary case (i.e. when (G, ρ) is involved), it is possible to utilize logical methods (see e.g. [1] for the occurring logical notions).

We regard that the elements of A are denoted by $a_1, a_2, \ldots, a_v$ (where $v = |A|$). The condition, stated in the Theorem, can be formulated as a conjunctive normal form $\mathfrak{N}$, expressing a truth function f. This function has $\binom{v}{2}$ variables w_{rs} (where $1 \leq r < s \leq v$) such that w_{rs} is true or false according as $\lambda(a_r) \neq \lambda(a_s)$ or $\lambda(a_r) = \lambda(a_s)$, respectively. We form, to any class L_i, the disjunction of the variables w_{rs} such that the edge $[a_r a_s]$ (exists in G and) belongs to L_i. We get $j+k$ elementary disjunctions (of nonnegated variables) in this manner; f is obtained by the formula $\mathfrak{N}$ which is the conjunction of these $j + k$ disjunctions.

It is known that a disjunctive normal form is often a more treatable representation of a truth function, than a conjunctive one. Therefore, if we continue the study of f, it may be useful to transform $\mathfrak{N}$ into a disjunctive normal form. Some methods for performing this are described in Chapter 3 of [1].

If a function f is analyzed, sometimes we may gain advantage from the idea that the variables w_{rs} are not independent of each other. Indeed, the equality is transitive, thus the formula

$$(\overline{w}_{rs} \& \overline{w}_{st}) \longrightarrow \overline{w}_{rt}$$

– or, equivalently, the formula

$$w_{rt} \longrightarrow (w_{rs} \vee w_{st})$$

– must be true for any choice of the subscripts r, s, t (where $\overline{w}_{rs}$ denotes the negation of w_{rs}).

§ 9.

In this section some examples will be studied. The semiautomata, analyzed in what follows, are mostly schemes of some automata occurring in previous articles.[4]

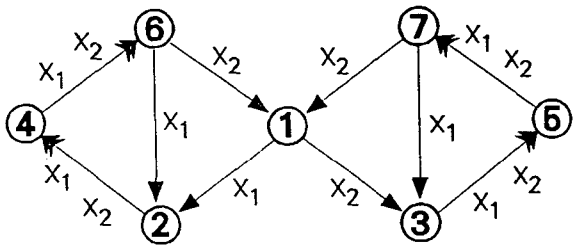

Fig. 1.

Example 1 *Put* $A = \{1, 2, \ldots, 7\}$ *and* $X = \{x_1, x_2\}$, *let* δ *be defined by Table 1 (see Fig. 1). Applying the first step of the Construction for this semiautomaton* $\mathbf{S}$, *we get the semiautomaton* $\mathbf{R} = (C, X, \delta_R)$ *seen in Fig. 2. (We write e.g. simply 2 instead of (2,2) in this figure.)* $\mathbf{R}$ *has 28 states, there are 21 proper pairs among the elements of* C. *There are four classes modulo* ε, *one class consists of the nonproper pairs. The proper pairs are distributed into three classes. One of these three classes is* $\{(1,2)\}$, *another class is*

$$\{(2,3),\ (4,5),\ (6,7)\}, \tag{4}$$

and the remaining 17 proper pairs belong to the third class. No class fulfils the conditions posed in Step 3 of the Construction. There is one class - namely (4) - which satisfies the conditions posed in Step 4. These facts mean that we have $j = 0, k = 1$ *and*

$$K_1 = \{(2,3),\ (4,5),\ (6,7)\}$$

in the present example.

[4]Compare the present Examples 1-3 with Example 3 in [4], Example 7 in [7], Example 6 in [8], respectively.

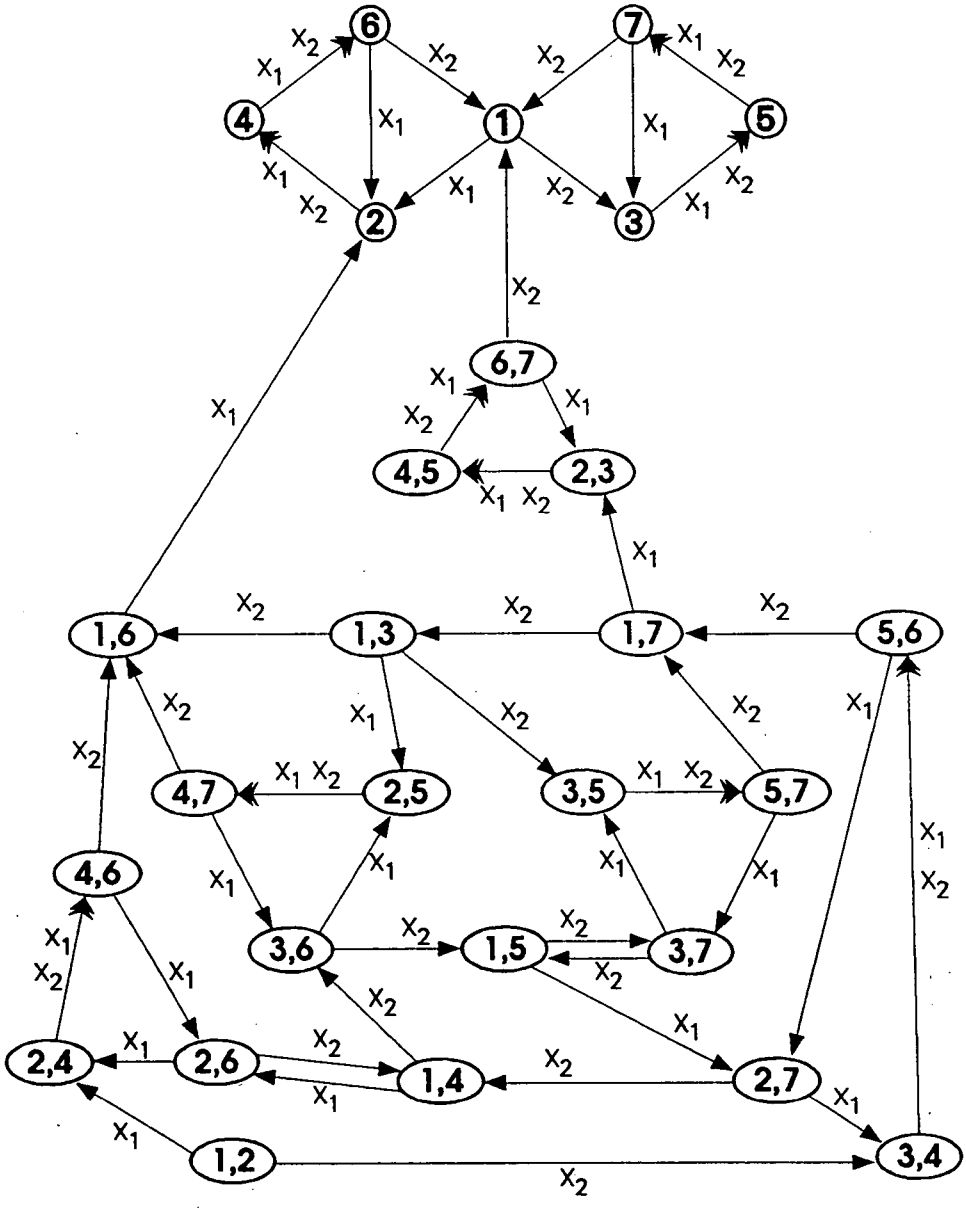
6
7
4
1
5
2
3
6,7
4,5
2,3
1,6
1,3
1,7
5,6
4,7
2,5
3,5
5,7
4,6
3,6
1,5
3,7
2,4
2,6
1,4
2,7
1,2
3,4
x1
x2

Fig. 2.

Fortunately, our discussion leads to a very simple situtation. The examination of the semiautomaton $\mathbf{S}$ terminates with constructing the graph G – seen in Fig. 3 – in which all the three edges belong to the same class L_1. Thus the criterion of the simplicity of an a-completion $\mathbf{A}_\lambda$ of $\mathbf{S}$ is

$$\lambda(2) \neq \lambda(3) \vee \lambda(4) \neq \lambda(5) \vee \lambda(6) \neq \lambda(7).$$

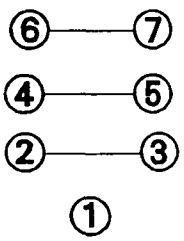

Fig. 3.

a	$\delta(a, x_1)$	$\delta(a, x_2)$
1	2	3
2	4	4
3	5	5
4	6	6
5	7	7
6	2	1
7	3	1

Table 1.

Example 2 *Put $A = \{1, 2, \ldots, 5\}$ and $X = \{x_1, x_2\}$, let δ be defined by Table 2 (see Fig. 4). In analogy to the preceding example, let $\mathbf{R}$ be constructed from this semiautomaton $\mathbf{S} = (A, X, \delta)$. (Some details can be left to the reader.) Among the 15 states of $\mathbf{R}$ there are 10 proper pairs. The number of classes mod ε, consisting of proper pairs, is three. Two of these classes fulfil the conditions of Step 3 of the Construction:*

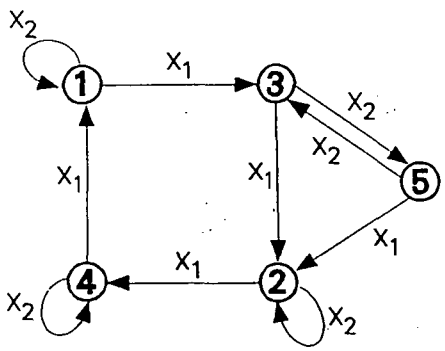

Fig. 4.

a	$\delta(a, x_1)$	$\delta(a, x_2)$
1	3	1
2	4	2
3	2	5
4	1	4
5	2	3

Table 2.

$$K_1 = \{(1,2),\ (3,4),\ (4,5)\},$$

$$K_2 = \{(1,3),\ (1,4),\ (1,5),\ (2,3),\ (2,4)\ (2,5)\}$$

(hence $j = 2$), and the third class

$$K_3 = \{(3,5)\}$$

satisfies the conditions of Step 4 (thus $k = 1$). The graph G has as many edges as possible, it is drawn in Fig. 5.

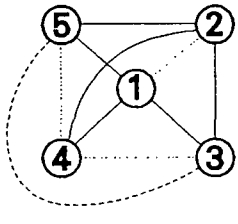

Fig 5.

Using the logical formalism considered in § 8, the criterion of the simplicity of an a-completion $\mathbf{A}_\lambda$ of $\mathbf{S}$ can be expressed by the conjunctive normal form

$$(w_{12} \vee w_{34} \vee w_{45})\&$$

$$\&(w_{13} \vee w_{14} \vee w_{15} \vee w_{23} \vee w_{24} \vee w_{25})\&w_{35}. \qquad (5)$$

Observe that $w_{35} \to (w_{13} \vee w_{15})$ and $w_{35} \to (w_{34} \vee w_{45})$ are identically true formulae (cf. the end of § 8). We can infer that the formula (5) is equivalent to w_{35}, consequently $\mathbf{A}_\lambda$ is simple if and only if $\lambda(3) \neq \lambda(5)$.

Although (5) was enough complicated, we were in the advantageous situation that we could obtain a remarkable simplification of (5) by utilizing the transitivity of the equality relation.

By analyzing Example 2, we see that the conclusion of Proposition 2 may hold for some a-completions $\mathbf{A}_\lambda$ of $\mathbf{S}$, but the supposition of Proposition 2 is false for each choice of λ. Hence the converse of Proposition 2 does not hold in general.

Example 3 *Put* $A = \{1, 2, \ldots, 10\}$, $X = \{x_1, x_2, x_3\}$, *let* δ *be defined by Table 3 (see Fig. 5 in [8]). Starting with this semiautomaton* $\mathbf{S}$, *let* $\mathbf{R}$ *and the equivalence relation* ε *be constructed. Consider the proper pairs*

a	$\delta(a, x_1)$	$\delta(a, x_2)$	$\delta(a, x_3)$
1	2	5	6
2	3	3	2
3	2	1	3
4	5	1	4
5	4	4	5
6	7	10	1
7	8	8	7
8	7	6	8
9	10	6	9
10	9	9	10

Table 3.

in C *only, then the number of elements in the 11 classes mod* ε *are: 24, 5, seven times 2, two times 1. By a further analysis we get that* $j = 1, k = 2$ *and the classes* K_1, K_2, K_3 *are:*

$$\begin{aligned} K_1 &= \{(1,6),(2,7),(3,8),(4,9),(5,10)\}, \\ K_2 &= \{(2,5),(3,4)\}, \\ K_3 &= \{(7,10),(8,9)\}. \end{aligned}$$

Thus the necessary and sufficient condition for the simplicity of an a-completion $\mathbf{A}_\lambda$ of $\mathbf{S}$ is the fulfilment of the logical formula

$$(w_{16} \vee w_{27} \vee w_{38} \vee w_{49} \vee w_{5,10}) \& (w_{25} \vee w_{34}) \& (w_{7,10} \vee w_{8,9}).$$

For the sake of completeness, let also the other classes of C mod ε be listed. They are:

$$\{(1,2),(2,6)\}, \{(1,7),(6,7)\}, \{(2,8),(3,7)\},$$
$$\{(2,10),(3,9)\}, \{(4,8),(5,7)\}, \{(2,3)\}, \{(7,8)\};$$

moreover, a class consisting of the remaining 24 proper pairs and a class to which the 10 nonproper pairs belong.

The section will be finished with two sequences of semiautomata. All the semiautomata $\mathbf{S}$, to be introduced in the sequel, have the property that, whenever a satate (i_1, i_2) of $\mathbf{R}$ is a proper pair, then $\{(i_1, i_2)\}$ is a separate class modulo ε.

Example 4 [5] *Choose a number* $v(\geq 2)$. *Put* $A = \{1, 2, \ldots, v\}, X = \{x_1, x_2\}$ *and let* δ *be defined in the following manner:*

$$\begin{aligned} \delta(1, x_1) &= 2, \\ \delta(i, x_1) &= 1 \quad \text{if} \quad 2 \leq i \leq v, \\ \delta(i, x_2) &= i + 1 \quad \text{if} \quad 1 \leq i \leq v - 1, \\ \delta(v, x_2) &= v. \end{aligned}$$

[5] This example is due to A. Nagy (personal communication).

We can observe that the proper pairs are indeed pairwise incongruent mod ε, furthermore, $j = 0, k = 1$ and $K_1 = \{(v-1, v)\}$. Thus the criterion of simplicity is $\lambda(v-1) \neq \lambda(v)$.

Example 5 [6] *Choose a number* $v(\geq 3)$. *Put* $A = \{1, 2, \ldots, v\}, X = \{x_1, x_2, \ldots, x_v\}$ *and let* δ *be defined in the following manner:*

$$\begin{aligned} \delta(1, x_h) &= h \quad \text{if} \quad 1 \leq h \leq v, \\ \delta(i, x_1) &= 1 \quad \text{if} \quad 2 \leq i \leq v, \\ \delta(i, x_h) &= i \quad \text{if} \quad 2 \leq i \leq v, \text{ and } 2 \leq h \leq v. \end{aligned}$$

We find that $j = 0$ and – because for a proper pair (i_1, i_2) the set $\{(i_1, i_2)\}$ satisfies the conditions of Step 4 of the Construction precisely if $2 \leq i_1 \leq v, 2 \leq i_2 \leq v$ are valid – we have $k = \binom{v-1}{2}$ and the criterion of simplicity is

$$|\{\lambda(2), \lambda(3), \lambda(4), \ldots, \lambda(v)\}| = v - 1.$$

For the reader who is interested in this subject, it can be recommended to study also the schemes of other automata occurring as examples in [7] and [8].

§ 10.

A semiautomaton $\mathbf{S} = (A, X, \delta)$ can be considered to be an object of complexity vn (where $n = |X|$ and – as earlier – $v = |A|$), since it can be characterized by a table having vn entries. The product vn is also a good (lower) estimate for the complexity of an a-completion of $\mathbf{S}$. From the view point of practical applications, that (semi-)automata are of primary interest for which n is remarkably smaller than v.

Start with a semiautomaton $\mathbf{S}$ and effectuate a construction of another procedure concerning $\mathbf{S}$. If the number of steps of the procedure is proportional to vn, then the procedure may be viewed economical as far as it is expectable. Such an optimal situation, however, is likely very infrequent. If the number of steps of a procedure is proportional to vn^β (with some exponent $\beta(> 1)$), then its complexity can be considered still as quite satisfactory. The procedures whose complexity is of order of magnitude $v^\alpha n^\beta$ (where $\alpha > 1$) are already worse ones, their profitableness decreases with the growth of α. At the other end of the scale, a procedure is not advantageous at all if its complexity cannot be estimated better than by an expression in which v occurs as an exponent.

Recall Proposition A, and consider the task that we are going to check whether or not the states of an automaton are pairwise indistinguishable. It is known[7] that two states a, b are distinguishable (if and) only if there is an input word p, fulfilling $\lambda(\delta(a, p)) \neq \lambda(\delta(b, p))$, such that the length of p does not exceed $v-2$. The number of input words whose length is at most $v-2$ equals

$$\frac{n^{v-1} - 1}{n - 1} (= 1 + n + n^2 + \cdots + n^{v-2}).$$

If we want to decide the simplicity of an automaton by using these ideas, we arrive at the following job:

[6] This example is due to F. Wettl (personal communication).

[7] See e.g. § 5 and § 12 in [3].

we draft a matrix of size $v \times ((n^{v-1}-1)/(n-1))$,

we fill the matrix with the output signs $\lambda(\delta(a,p))$ as its entries, and

we examine the existence of two rows of the matrix that are from place to place coinciding.

The complexity of this process depends exponentially on v, consequently, it is not in the least economical.

The method, based in §§ 5-6 of this paper, is such an improvement of the "rough" application of Proposition A that its complexity remains already under polynomial bounds. The order magnitude of the semiautomaton **R** is $v(v+1)n/2$, this quantity is approximately proportional to v^2n. Although the number 2 (as exponent of v) is not quite reassuring, the author is afraid that it cannot be diminished notably (unless we restrict our attention to one or another particular class of semiautomata).

A known algorithm due to Tarjan (see [15]) shows that the classes of the mutual accessibility relation in directed graphs can be determined so that the complexity depends linearly on the number of vertices (if the ratio of the edge number and the vertex number is bounded); consequently, the computational complexity of our Construction is not increased in Steps 2-5 (in comparison to the complexity of Step 1).

§ 11.

In this final section further comments will be done concerning the Construction (in § 5), the Theorem (at the end of § 6) and the handling of the question by logical tools (see § 8).

It is not quite hopeless that the method (elaborated in §§ 5-8) can be refined into a more economical process under certain particular conditions. This subject will be concerned in the first three problems to be raised at once (they are rather heuristical than exact ones). The study of these problems is desirable primarily within the class of strongly connected semiautomata, because a reduction of the general question of the simplicity of automata to the strongly connected case is already known (see [5], [6]).

Problem 1. Find semiautomaton classes such that, for the elements of a class, the graph (G,ρ) can be obtained by some remarkably easier way, than through constructing the semiautomaton **R**.

Problem 2. Study circumstances under which the truth function f - assigned to the graph (G,ρ) - admits an easy discussion. (f is, of course, easily treatable if it is got by a short formula. Beside this case, Problem 2 concerns whether the following methods can be utlized adavantageously: conversion of a conjunctive normal form into a disjunctive one, and/or use of the consequences of the transitivity of the equality relation.)

Consider again the partitioned graph (G,ρ) obtained in Step 5 of the Construction. Denote the number of the non-adjacent proper vertex pairs, i.e. the quantity

$$\binom{v}{2} - (|L_1| + |L_2| + \cdots + |L_{j+k}|),$$

by $\eta(G)$. The quotient

$$\frac{\max(|L_1|, |L_2|, \ldots, |L_{j+k}|, \eta(G))}{\binom{v}{2}} \tag{6}$$

can be viewed as a measure of in what degree (G, ρ) is perspicuous. The value (6) is clearly between $1/(j+k+1)$ and 1.

Problem 3. Find semiautomaton classes such that, for the elements of a class, the value of the expression (6) is near to one.

The last problem will be devoted to the connection between the general criterion of simplicity, asserted as the Theorem, and the known criterion for the simplicity of *autonomous* Moore automata, having been stated in [4]. The latter result can be formulated as follows:

Proposition D. ([4], Proposition 6). *Let* $\mathbf{S}$ *be an autonomous Moore semiautomaton. An a-completion* $\mathbf{A}_\lambda$ *of* $\mathbf{S}$ *is simple exactly if* λ *fulfils the following conditions:*

(i) *each cycle is primitive,*[8]
(ii) *the cycles are pairwise non-isomorphic,*
(iii) *whenever* $\delta(a,x) = \delta(b,x)$ *for a proper state pair* (a,b) *then* $\lambda(a) \neq \lambda(b)$.

Let condition (II) of the Theorem be applied for an autonomous semiautomaton. It is then almost obvious to see that condition (iii) is necessary for the simplicity of an $\mathbf{A}_\lambda$. In the other respects, however, it appears no immediate possibility for deriving Proposition D from the Theorem.

Problem 4. Show that the necessity of the conditions (i), (ii) and the sufficiency of (i) & (ii) & (iii) are consequences of the Theorem when (particularly) an autonomous semiautomaton is considered.

References

[1] Ádám, A., *Truth functions and the problem of their realization by two-terminal graphs,* Akadémiai Kiadó, Budapest, 1968.

[2] Ádám, A., On the question of description of the behaviour of finite automata, *Studia Sci. Math. Hungar.* 13 (1978), 105-124.

[3] Ádám, A., On certain partitions of finite directed graphs and of finite automata, *Acta Cybernetica* (Szeged) 6 (1984), 331-346.

[4] Ádám. A., On the congruences of finite autonomous Moore automata, *Acta Cybernetica* (Szeged) 7 (1986), 259-279.

[5] Ádám, A., On simplicity-critical Moore automata, I, *Acta Math. Hungar.* 52 (1988), 165-174.

[8] See [4], pp. 261-262 for the definition of primitivity.

[6] Ádám, A., On simplicity-critical Moore automata, II, *Acta Math. Hungar.* 54 (1989), 291–296.

[7] Ádám, A. and Babcsányi, I., Results and problems on strongly connected Moore automata, *Studia Sci. Math. Hungar.* (To appear)

[8] Ádám, A. and Wettl, F., On two realizability questions concerning strongly connected Moore automata, *Publ. Math.* (Debrecen). (To appear)

[9] Babcsányi, I., Nagy, A. and Wettl, F., Indistinguishable state pairs in strongly connected Moore automata, *Pure Math. and Applications, A* 2 (1991), 15-24.

[10] Gécseg, F. and Peák, I., *Algebraic theory of automata,* Akadémiai Kiadó, Budapest, 1972.

[11] Harary, F., *Graph theory,* Addison-Wesley, Reading, 1969.

[12] Harary, F., *Teorija grafov,* Mir. Moskva, 1973. (Russian translation of [11]).

[13] Harary, F., Norman, R.Z. and Cartwright, D., *Structural models: An introduction to the theory of directed graphs,* Wiley, New York, 1965.

[14] Harary, F., Norman, R.Z. and Cartwright, D., *Introduction à la théorie des graphes orientés, Modèles structureaux,* Dunod, Paris, 1968. (French translation of [13]).

[15] Tarjan, R., Depth-first search and linear graph algorithms, *SIAM J. Comput.* 1 (1972), 146-160.

Received May 8, 1992

Acta Cybernetica, Vol. 10, Nr. 4, Szeged, 1992

On a special composition of tree automata

B. Imreh*†

In the theory of finite automata it is an interesting problem to describe such systems from which any automaton can be built under a given composition and isomorphic embedding as representation. Such systems are called isomorphically complete with respect to the considered composition. In particular, it is important to characterize those compositions for which there are finite isomorphically complete systems. In the works [1], [2] necessary conditions are given for the existence of finite isomorphically complete systems with respect to the classical automata and tree automata, respectively. In both cases it turned out that the existence of a finite isomorphically complete system yields the unboundedness of the feedback dependency of the composition. It is unknown yet whether this condition is sufficient. So it is interesting to investigate such compositions for which there are finite isomorphically complete systems. In [4] such a composition was introduced. Here we generalize this notion of composition to tree automata and give a necessary and sufficient condition of the isomorphic completeness. For this reason we recall some notions from [3] and [5].

By a set of *operational symbols* we mean a nonempty union $\Sigma = \Sigma_0 \bigcup \Sigma_1 \bigcup \ldots$, where Σ_m $(m = 0, 1, \ldots)$ are pairwise disjoint sets of symbols. For any $m \geq 0$, the set Σ_m is called the set of *m-ary operational symbols*. It is said that the *rank* or *arity* of a symbol $\sigma \in \Sigma$ is m if $\sigma \in \Sigma_m$. Now let a set Σ of operational symbols and a set R of nonnegative integers be given. R is called the *rank-type* of Σ if for any integer $m \geq 0$, $\Sigma_m \neq \emptyset$ if and only if $m \in R$. Next we shall work under a fixed rank-type R.

Now let Σ be a set of operational symbols with rank-type R. By a *Σ-algebra* $\mathcal{A}$ we mean a pair consisting of a nonempty set A and a mapping that assigns to every operational symbol $\sigma \in \Sigma$ an m-ary operation $\sigma^{\mathcal{A}} : A^m \to A$, where the arity of σ is m. The set A is called the *set of elements of* $\mathcal{A}$ and $\sigma^{\mathcal{A}}$ is the *realization of* σ *in* $\mathcal{A}$. The mapping $\sigma \to \sigma^{\mathcal{A}}$ will not be mentioned explicitly, but we write $\mathcal{A} = (A, \Sigma)$. It is said that a Σ-algebra $\mathcal{A}$ is *finite* if A is finite, and it is of *finite type* if Σ is finite. By *a tree automaton* we mean a finite algebra of finite type. Finally, it is said that the *rank-type of a tree automaton* $\mathcal{A} = (A, \Sigma)$ is R if the rank-type of Σ is R.

Now let us denote by $\mathcal{U}_R$ the class of all tree automata with rank-type R. A composition of tree automata from $\mathcal{U}_R$ can be represented as a network in which each vertex denotes a tree automaton and the actual operation of a tree automaton may depend only on those automata which have direct connection to the given one.

In order to define this notion of composition let D denote an arbitrary nonempty fixed set of finite directed graphs. Let $\mathcal{A} = (A, \Sigma) \in \mathcal{U}_R$ and $\mathcal{A}_j = (A_j, \Sigma^j) \in \mathcal{U}_R$ $(j = 1, \ldots, n)$. Moreover, take a family Ψ of mappings

*Department of Informatics, A. József University, Árpád tér 2, Szeged 6720, Hungary

†This paper was supported by Hungarian Foundation for Scientific Research (OTKA), Grant 2035.

$$\Psi_{mj} : (A_1 \times \ldots \times A_n)^m \times \Sigma_m \to \Sigma_m^j \qquad (m \in R,\ 1 \leq j \leq n) .$$

It is said that the tree automaton $\mathcal{A}$ is a $\mathcal{D}$-*product* of $\mathcal{A}_j$ $(j = 1, \ldots, n)$ with respect to Ψ if the following conditions are satisfied:

(i) $A = \prod_{j=1}^{n} A_j$,

(ii) there exists a graph $D = (\{1, \ldots, n\}, E)$ in $\mathcal{D}$ such that for any $m \in R$, $j \in \{1, \ldots, n\}$ and

$$((a_{11}, \ldots, a_{1n}), \ldots, (a_{m1}, \ldots, a_{mn})) \in A^m$$

the mapping Ψ_{mj} is independent of the elements a_{ts} $(t = 1, \ldots, m)$ if $(s, j) \notin E$,

(iii) for any $m \in R$, $\sigma \in \Sigma_m$ and $((a_{11}, \ldots, a_{1n}), \ldots, (a_{m1}, \ldots, a_{mn})) \in A^m$,

$$\sigma^{\mathcal{A}}((a_{11}, \ldots, a_{1n}), \ldots, (a_{m1}, \ldots, a_{mn})) = (\sigma_1^{\mathcal{A}_1}(a_{11}, \ldots, a_{m1}), \ldots, \sigma_n^{\mathcal{A}_n}(a_{1n}, \ldots, a_{mn}))$$

where

$$\sigma_j = \Psi_{mj}((a_{11}, \ldots, a_{1n}), \ldots, (a_{m1}, \ldots, a_{mn}), \sigma) \qquad (j = 1, \ldots, n) .$$

We shall use the notation

$$\prod_{j=1}^{n} \mathcal{A}_j(\Sigma, \Psi, D)$$

for the product introduced above and sometimes we shall indicate only those variables of ψ_{mj} on which it may depend.

Now let $\mathcal{B}$ be a system of tree automata from $\mathcal{U}_R$. It is said that $\mathcal{B}$ is *isomorphically complete for* $\mathcal{U}_R$ *with respect to the* $\mathcal{D}$-*product* if any tree automaton from $\mathcal{U}_R$ can be embedded isomorphically into a $\mathcal{D}$-product of tree automata from $\mathcal{B}$.

The first characterization of isomorphically complete systems of tree automata was given in [5] with respect to the Gluskov-type product, which can be defined considering the set of finite directed complete graphs as possible networks. Now taking the set of the n-dimensional hyper cubes $(n = 2, 3, \ldots)$ as possible networks, we prove that this cube-product is equivalent to the Gluskov-type product with respect to the isomorphic completeness. For this purpose we need some preparation.

Let $n \geq 2$ be an arbitrary integer. Let us consider the n-dimensional hyper cube. The set of the vertices of this hyper cube is $S_n = \{(s_1, \ldots, s_n) : s_i \in \{0, 1\}\ (i = 1, \ldots, n)\}$. Define the mapping λ_n on the set S_n as follows: for any vector $(s_1, \ldots, s_n)$

$$\lambda_n(s_1, \ldots, s_n) = 1 + \sum_{t=1}^{n} s_t \cdot 2^{n-t} .$$

Then λ_n is a one-to-one mapping of S_n onto the set $\{1, \ldots, 2^n\}$.

Let us form the directed graph $D_n^* = (\{1, \ldots, 2^n\}, V_n)$, where for any $1 \leq i, j \leq 2^n$, $(i, j) \in V_n$ if and only if $\lambda_n^{-1}(i)$ is adjacent to $\lambda_n^{-1}(j)$. For any $u \in \{1, \ldots, 2^n\}$

let us denote by $J_u^{(n)}$ the set of all ancestors of u in D_n^*. It is obvious that $\lambda_n^{-1}(u) = (s_1, \ldots, s_n)$ is adjacent to a vertex $(r_1, \ldots, r_n)$ if and only if there exists an index $1 \leq i \leq n$ such that $r_i = 1 - s_i$ and $r_j = s_j$ if $1 \leq j \leq n$ and $i \neq j$. Therefore, $|J_u^{(n)}| = n$, i.e. each vertex of D_n^* has exactly n ancestors. On the other hand, it is easy to see that

if $1 \leq u \leq 2^{n-1}$, then u has one ancestor in the set $\{2^{n-1}+1, \ldots, 2^n\}$ and $n-1$ ancestors in the set $\{1, \ldots, 2^{n-1}\}$,

if $2^{n-1} < u \leq 2^n$, then u has one ancestor in the set $\{1, \ldots, 2^{n-1}\}$ and $n-1$ ancestors in the set $\{2^{n-1}+1, \ldots, 2^n\}$.

Now let us suppose that $n > 2$ and consider the graphs D_n^* and D_{n-1}^*. Then using the above observation, one can prove the following equalities:

$$J_u^{(n-1)} = J_u^{(n)} \setminus \{u + 2^{n-1}\} \quad \text{if} \quad 1 \leq u \leq 2^{n-1} \quad \text{and} \tag{1}$$

$$J_{u-2^{n-1}}^{(n-1)} = \{v - 2^{n-1} : v \in (J_u^{(n)} \setminus \{u - 2^{n-1}\})\} \quad \text{if} \quad 2^{n-1} < u \leq 2^n \ . \tag{2}$$

Now we are ready to prove our statement.

Theorem 0.1 *Let $\mathcal{D}^* = \{D_n^* : n = 2, 3, \ldots\}$. A system $\mathcal{C} \subseteq \mathcal{U}_R$ of tree automata is isomorphically complete for $\mathcal{U}_R$ with respect to the $\mathcal{D}^*$-product if and only if $\mathcal{C}$ contains a tree automaton $\mathcal{A} = (A, \Sigma)$ which has two different states a, b and for any $m \in R$, $(u_1, \ldots, u_m) \in \{a, b\}^m$, $u \in \{a, b\}$ there exists an m-ary operation $\sigma \in \Sigma$ with $\sigma^{\mathcal{A}}(u_1, \ldots, u_m) = u$.*

Proof. If $R = \{0\}$, then the validity of our statement can be proved easily. Now let us suppose that $R \neq \{0\}$. Then the necessity follows from the work [5].

In order to prove the sufficiency, first let us define the sequence of matrices $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}, \ldots$ as follows:

$$\mathbf{A}^{(1)} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} \qquad \mathbf{A}^{(n+1)} = \begin{pmatrix} \mathbf{A}^{(n)} & \mathbf{A}^{(n)} \\ \mathbf{A}^{(n)} & \bar{\mathbf{A}}^{(n)} \end{pmatrix} \qquad (n = 1, 2, \ldots) \ ,$$

where $\bar{\mathbf{A}}^{(n)}$ is defined by $\bar{a}_{ij}^{(n)} = 1 - a_{ij}^{(n)}$ $(1 \leq i \leq 2^{n+1};\ 1 \leq j \leq 2^n)$ in the partitioned matrix.

We shall show that for any $n \geq 2$ and $1 \leq u \leq 2^n$ the n-tuples $(a_{ti_1}^{(n)}, \ldots, a_{ti_n}^{(n)})$ $(t = 1, \ldots, 2^n)$ are pairwise different, where $\{i_1, \ldots, i_n\} = J_u^{(n)}$.

We proceed by induction on n. The case $n = 2$ can be checked easily. Now let $n > 2$ and assume that the statement is valid for $n-1$. Let $1 \leq u \leq 2^n$ be arbitrary and $J_u^{(n)} = \{i_1, \ldots, i_n\}$. Let us suppose that $i_v < i_w$ if $v < w$. If the desired n-tuples are pairwise not different, then there are indices j, k with $1 \leq j < k \leq 2^n$ such that

$$(a_{ji_1}^{(n)}, \ldots, a_{ji_n}^{(n)}) = (a_{ki_1}^{(n)}, \ldots, a_{ki_n}^{(n)}) \ . \tag{3}$$

Now we distinguish three cases.

Case 1. Let us suppose that $1 \le j < k \le 2^{n-1}$. If $1 \le u \le 2^{n-1}$, then $n-1$ ancestors of u are in the set $\{1,\dots,2^{n-1}\}$ and the nth ancestor is $u+2^{n-1}$. Therefore, by the ordering of $J_u^{(n)}$, $i_n = u+2^{n-1}$. Then, by (1), $J_u^{(n-1)} = \{i_1,\dots,i_{n-1}\}$ and by the definition of $\mathbf{A}^{(n)}$,

$$(a_{ji_1}^{(n-1)},\dots,a_{ji_{n-1}}^{(n-1)}) = (a_{ji_1}^{(n)},\dots,a_{ji_{n-1}}^{(n)}) =$$

$$(a_{ki_1}^{(n)},\dots,a_{ki_{n-1}}^{(n)}) = (a_{ki_1}^{(n-1)},\dots,a_{ki_{n-1}}^{(n-1)})$$

which contradicts our induction assumption.

If $2^{n-1} < u \le 2^n$, then $n-1$ ancestors of u are in the set $\{2^{n-1}+1,\dots,2^n\}$ and the nth ancestor is $u-2^{n-1}$. Therefore, $i_1 = u-2^{n-1}$. Let $w_t = i_t - 2^{n-1}$ $(t=2,\dots,n)$. Then by (2), $J_{i_1}^{(n-1)} = \{w_2,\dots,w_n\}$. But then using the equality (3) and the definition of $\mathbf{A}^{(n)}$, we obtain that

$$(a_{jw_2}^{(n-1)},\dots,a_{jw_n}^{(n-1)}) = (a_{kw_2}^{(n-1)},\dots,a_{kw_n}^{(n-1)})$$

which contradicts our induction assumption.

Case 2. Assume that $2^{n-1} < j < k \le 2^n$.

Let $r = j-2^{n-1}$, $s = k-2^{n-1}$. Then $1 \le r < s \le 2^{n-1}$. On the other hand, by the construction of $\mathbf{A}^{(n-1)}$, from (3) it follows that $(a_{ri_1}^{(n)},\dots,a_{ri_n}^{(n)}) = (a_{si_1}^{(n)},\dots,a_{si_n}^{(n)})$ which yields a contradiction in the same way as in Case 1.

Case 3. Let us suppose that $1 \le j \le 2^{n-1} < k \le 2^n$.

If $1 \le u \le 2^{n-2}$, then by (1), $i_n = u+2^{n-1}$, $i_{n-1} = u+2^{n-2}$ and $J_u^{(n-2)} = \{i_1,\dots,i_{n-2}\} \subseteq \{1,\dots,2^{n-2}\}$. Since $i_n = u+2^{n-1}$, by the definition of $\mathbf{A}^{(n)}$ and (3), we obtain $a_{ju}^{(n)} = a_{ji_n}^{(n)} = a_{ki_n}^{(n)} = a_{ku}^{(n)}$. By (3), $a_{ji_{n-1}}^{(n)} = a_{ki_{n-1}}^{(n)}$, which results that $k \ne j+2^{n-1}$. Now let $r = k-2^{n-1}$. Then $1 \le r \le 2^{n-1}$. Since $1 \le u \le 2^{n-2}$ and $2^{n-2} < i_{n-1} \le 2^{n-1}$, by the construction of $\mathbf{A}^{(n-1)}$, we obtain $a_{ku}^{(n)} = a_{ru}^{(n)}$, $a_{ki_{n-1}}^{(n)} = 1 - a_{ri_{n-1}}^{(n)}$. But then $a_{ji_{n-1}}^{(n)} = 1 - a_{ri_{n-1}}^{(n)}$. On the other hand, $1 \le u \le 2^{n-2}$, $u+2^{n-2} = i_{n-1}$, $a_{ju}^{(n)} = a_{ru}^{(n)}$, $1 \le j,r \le 2^{n-1}$ yield that $a_{ji_{n-1}}^{(n)} = a_{ri_{n-1}}^{(n)}$ which is a contradiction.

If $2^{n-2} < u \le 2^{n-1}$, then on the bases of (1) and (2), $i_n = u+2^{n-1}$, $i_1 = u-2^{n-2}$ and $\{i_2,\dots,i_{n-1}\} \subseteq \{2^{n-2}+1,\dots,2^{n-1}\}$. But then, by (3) and the definition of $\mathbf{A}^{(n)}$, $a_{ju}^{(n)} = a_{ku}^{(n)}$, which yields $k \ne j+2^{(n-1)}$. Let $r = k-2^{n-1}$. By the construction of $\mathbf{A}^{(n-1)}$, $a_{ru}^{(n)} = 1 - a_{ku}^{(n)}$, and so, $a_{ru}^{(n)} = 1 - a_{ju}^{(n)}$. On the other hand, by (3), $a_{ki_1}^{(n)} = a_{ji_1}^{(n)}$, and so, by the construction of $\mathbf{A}^{(n-1)}$, $a_{ri_1}^{(n)} = a_{ji_1}^{(n)}$. Since $i_1 + 2^{n-2} = u$ and $1 \le j,r \le 2^{n-1}$, by the construction of $\mathbf{A}^{(n-1)}$, we obtain that the last equality yields $a_{ru}^{(n)} = a_{ju}^{(n)}$ which is a contradiction.

If $2^{n-1} < u \le 2^n$, then $i_1 = u-2^{n-1}$. Let $w_{t-1} = i_t - 2^{n-1}$ $(t=2,\dots,n)$ and $w_n = i_1 + 2^{n-1} = u$. Then by (1) and (2), $J_{i_1}^{(n)} = \{w_1,\dots,w_n\}$. On the other

hand, by (3) and the definition of $\mathbf{A}^{(n)}$, we obtain the equality $(a^{(n)}_{jw_1},\dots,a^{(n)}_{jw_n}) = (a^{(n)}_{kw_1},\dots,a^{(n)}_{kw_n})$. Since $1 \le i_1 \le 2^{n-1}$, we have traced back the considered case to the above treated ones.

Now let us suppose that C contains a tree automaton $\mathcal{A} = (A,\Sigma)$ satisfying the conditions of our Theorem with the elements a, b. Without loss of generality we may assume that $a = 0$ and $b = 1$. Furthermore, for any $m \in R$, $(u_1,\dots,u_m) \in \{0,1\}^m$, $u \in \{0,1\}$ let us denote by $\sigma_{u_1,\dots,u_m,u}$ an operational symbol from Σ_m for which $\sigma^{\mathcal{A}}_{u_1,\dots,u_m,u}(u_1,\dots,u_m) = u$ holds.

Now let $\mathcal{B} = (\{b_1,\dots,b_w\},\Sigma')$ be an arbitrary tree automaton. Choose an integer $n \ge 2$ such that $w \le 2^n$. Let μ be a one to one mapping of $\{b_1,\dots,b_w\}$ onto the first w rows of the matrix $\mathbf{A}^{(n)}$ defined by $\mu(b_k) = (a^{(n)}_{k1},\dots,a^{(n)}_{k2^n})$ $(k = 1,\dots,w)$. Denote by S the set $\{\mu(b_k) : k = 1,\dots,w\}$. Let $1 \le u \le 2^n$ be arbitrary. We know that the n-tuples $(a^{(n)}_{ti_1},\dots,a^{(n)}_{ti_n})$ $(t = 1,\dots,2^n)$ are pairwise different, where $\{i_1,\dots,i_n\} = J^{(n)}_u$. But then there is a one to one mapping τ_u for which $\tau_u(a^{(n)}_{ti_1},\dots,a^{(n)}_{ti_n}) = b_t$ $(t = 1,\dots,w)$. Let us consider these mappings τ_u for any $1 \le u \le 2^n$.

Take the D^*-product $\tilde{\mathcal{A}} = \prod_{j=1}^{2^n} \mathcal{A}(\Sigma',\Psi,D^*_n)$, where the family Ψ of mappings is defined as follows:

For any $0 \ne m \in R$, $\sigma \in \Sigma'_m$, $1 \le u \le 2^n$ and $s_t = (s_{t1},\dots,s_{t2^n}) \in S$ $(t = 1,\dots,m)$,

$$\Psi_{mu}(s_1,\dots,s_m,\sigma) = \sigma_{s_{1i_1},\dots,s_{mi_n},a^{(n)}_{ku}}$$

where $\sigma^{\mathcal{B}}(\tau_u(s_{1i_1},\dots,s_{1i_n}),\dots,\tau_u(s_{mi_1},\dots,s_{mi_n})) = b_k$.

If $0 \in R$, $\sigma \in \Sigma'_0$ and $\mu(\sigma^{\mathcal{B}}) = (a^{(n)}_{k1},\dots,a^{(n)}_{k2^n})$, then

$$\Psi_{mu}(\sigma) = \sigma^{\mathcal{A}}_{a^{(n)}_{ku}}$$

For any $m \in R$, $\sigma \in \Sigma'$, $1 \le u \le 2^n$ and $((u_{11},\dots,u_{12^n}),\dots(u_{m1},\dots,u_{m2^n})) \in \{A^{2^n}\}^m \setminus S^m$, Ψ_{mu} is defined arbitrarily in accordance with the definition of the D^*-product.

It is easy to see that the mappings Ψ_{mu} are well-defined, and so, we obtain a D^*-product. On the other hand, one can prove that μ is an isomorphism of $\mathcal{B}$ into $\tilde{\mathcal{A}}$. Therefore, $\{\mathcal{A}\}$ is an isomorphically complete system for U_R with respect to the D^*-product, which completes our proof.

Remark. Characterization of the isomorphically complete systems with respect to the Gluskov-type product (see [5]) is the same as the characterization in our Theorem. So this two kind of products are equivalent with respect to the isomorphically complete systems.

Acknowledgement. The author is very greatful to Professor F. Gécseg for his helpful remarks which greatly improved this paper.

References

[1] F. Gécseg and B. Imreh, Finite isomorphically complete systems, *Discrete Applied Mathematics*, **36** (1992),307-311.

[2] F. Gécseg and B. Imreh, On finite isomorphically complete systems of tree automata, Acta Sci. Math., to appear

[3] F. Gécseg and M. Steinby, *Tree automata*, Akadémiai Kiadó, Budapest,1984.

[4] B. Imreh, On complete systems of automata, in: Proc. of the 2nd InternationalColloquium on Words, Languages and Combinatorics, to appear.

[5] M. Steinby, *On the structure and realizations of tree automata*, in Second Coll. sur les Arbres an Algèbre et en Programmation (Lille, 1979), 235-248.

Received December 4, 1992

Acta Cybernetica, Vol. 10, Nr. 4, Szeged, 1992

Regularizing context-free languages by AFL operations: concatenation and Kleene closure

J. Dassow * A. Mateescu †‡ G. Paun §¶ A. Salomaa ‖

Abstract

We consider the possibility to obtain a regular language by applying a given operation to a context-free language. Properties of the family of context-free languages which can be "regularized" by concatenation with a regular set or by Kleene closure are investigated here: size, hierarchies, characterizations, closure, decidability.

1 Introduction

The core of formal language theory is the study of the Chomsky hierarchy, especially of families of regular and of context-free languages. An important problem in this context is to understand the differences between "regularity" and "context-freeness". The question is approached, explicitly or implicitly, in many papers.

Here we follow [2], [3], [4], [7] and consider this problem in relation with operations with languages. Usually, the main topic dealt with when investigating operations with languages is the closure of various families (how much an operation can "complicate" a language). A dual natural question is "how much an operation can simplify languages in a given family". In particular, we are interested in transforming in this way context-free languages into regular languages.

Similar problems are investigated in [2], [4], whereas [3], [7] consider numerical measures of non-regularity of context-free languages and the influence of various operations on them.

Here we investigate the possibility of obtaining a regular language starting from a context-free language and using one of the six AFL operations: union, concatenation, intersection - all by regular sets -, Kleene closure, morphisms and inverse

*Otto von Guericke University of Magdeburg, Department of Computer Science, PSF 4120, O - 3010 Magdeburg, Germany

†Institute of Mathematics of the Romanian Academy of Sciences, Str. Academiei 14, 70109 Bucuresti, Romania

‡Research supported by the Academy of Finland, grant nr 11281

§Institute of Mathematics of the Romanian Academy of Sciences, Str. Academiei 14, 70109 Bucuresti, Romania

¶Research supported by Alexander von Humboldt Foundation

‖Academy of Finland and University of Turku, Department of Mathematics, SF - 20500 Turku 50, Finland

morphisms. We enter into details only for the right and left concatenation and for Kleene *, namely we study the properties of families of context-free languages which can lead to regular languages by left/right concatenation with regular sets of by Kleene *.

2 Notations

For an alphabet V, we denote by V^* the free monoid generated by V under the operation of concatenation; the null element of V^* is denoted by λ and $|x|$ denotes the length of $x \in V^*$. For $x \in V^*, a \in V$, we denote by $|x|_a$ the number of occurrences in x of the symbol a. We denote also by REG, LIN, CF the families of regular, linear and context-free languages.

For a language L we denote by $Pref(L), Suf(L), Sub(L)$ the sets of prefixes, suffixes, respectively subwords of strings in L.

The main problem of this paper is the following: given a language $L \in CF$ and an operation with languages, can we use this operation in such a way to obtain a regular language starting from L ?

In this form, the question is trivial for most AFL operations. For instance, for all context-free languages $L \subseteq V^*$, the languages

(i) $L \cup V^* = V^*$,
(ii) $h(L)$ for all $h : V^* \longrightarrow \{a\}^*$,
(iii) $L \cap R$ for all finite languages R,
(iv) $h^{-1}(L)$ for all $h : \{a\}^* \longrightarrow V^*$,

are regular. The question is not trivial for concatenation and Kleene closure:

(i) Concatenating (on the left side) the non-regular language

$$L_1 = \{a^n b^m \mid 1 \leq n \leq m\}$$

with

$$R = \{a^p \mid p \geq 1\},$$

we obtain a regular language, but no right or left concatenation of

$$L_2 = \{a^n b^n \mid n \geq 1\}$$

with a non-empty set will give a regular language (if $RL_2 \in REG$, for some R, then take $x \in R$ and intersect RL_2 with xa^*b^*; the obtained language is not regular, hence RL_2 is not regular, a contradiction).

(ii) For the above language L_2, the language L_2^* is not regular, but for

$$L_3 = L_2 \cup \{a, b\}$$

we have

$$L_3^* = \{a, b\}^*,$$

which is regular.

Thus, we are led to consider the families

$CL = \{L \in CF \mid$ there is $R \in REG, R \neq \emptyset$, such that $RL \in REG\}$,
$CR = \{L \in CF \mid$ there is $R \in REG, R \neq \emptyset$, such that $LR \in REG\}$,
$K = \{L \in CF \mid L^* \in REG\}$,
$K_n = \{L \in CF \mid$ there is $1 \leq m \leq n$ such that $\bigcup_{i=1}^{m} L^i \in REG\}$, for $n \geq 1$.

We shall investigate here only the families $CL, K, K_n, n \geq 1$; the results for CL are true also for CR, with obvious modifications.

3 The size of the families introduced above

The next relations follow from definitions.

Lemma 3.1 *(i)* $REG \subseteq CL \subseteq CF$,
(ii) $REG \subseteq K \subseteq CF$,
(iii) $REG = K_1 \subseteq K_2 \subseteq \ldots \subseteq CF$.

Lemma 3.2 $K_n \subseteq K$, *for all* $n \geq 1$.

Proof. Take $L \in K_n$. There is $m \leq n$ such that $\bigcup_{i=1}^{m} L^i \in REG$. Clearly, $L^* = (\bigcup_{i=1}^{m} L^i)^*$, hence also L^* is regular, that is $L \in K$. □

All these inclusions are proper.

Theorem 3.3 $REG \subset CL \subset CF$.

Proof. The language L_1 in the previous section is in CL but it is not regular, whereas the language L_2 in the previous section is not in $CL \cup CR$. □

Lemma 3.4 *(i) If an arbitrary language* $L \subseteq V^*$ *satisfies, for some* $k \geq 0$, *the relation* $V^k \subseteq L$, *then* $V^*L \in REG$. *In particular, if* $\lambda \in L$, *then* $V^*L \in REG$.

(ii) If an arbitrary language $L \in V^*$ *satisfies, for some* $k_1, k_2 \geq 0, k_1, k_2$ *relatively prime, the relation* $V^{k_1} \cup V^{k_2} \subseteq L$, *then* $L^* \in REG$.

Proof. (i) Under the previous conditions, we obtain

$$V^*L = V^*L_k,$$

for $L_k = \{x \in L \mid |x| \leq k\}$.

The inclusion $\subseteq$ is obvious. Conversely, take $x, y \in V^*L$, $x \in V^*, y \in L$. If $|y| \leq k$, then $y \in L_k, xy \in V^*L_k$. If $|y| > k$, then $y = y_1y_2, |y_2| = k$. As $xy_1 \in V^*$, we have again $xy = xy_1y_2 \in V^*L_k$.

The language L_k is finite, hence assertion (i) follows.

(ii) Note that, because k_1, k_2 are relatively prime, there exists $m_0, m_0 \in \mathbf{N}$, such that for any $n \geq m_0$ there are $i, j \in \mathbf{N}$ with $n = ik_1 + jk_2$. Thus L^* contains all words w such that $|w| \geq m_0$, hence $V^* - L^*$ is a finite set; consequently, L^* is regular. □

Corollary 3.5 CL *is incomparable with* LIN.

Proof. The above considered language L_2 proves the relation $LIN - CL \neq \emptyset$.

Conversely, take the Dyck language D over $\{a, b\}$. We have $D \in CF - LIN$. It contains the string λ, hence $D \in CL$ and $CL - LIN \neq \emptyset$ too. □

Corollary 3.6 *For every context-free language* $L, L \subseteq V^*$, *either* L *or* $V^* - L$ *is in* CL.

Proof. Obvious, as one of L and $V^* - L$ contains the null string.

Theorem 3.7 $REG \subset K \subset CF$.

Proof. For all $L \in CF, L \subseteq V^*$, the language $L' = L \cup V$ is in K, as $(L \cup V)^* = V^*$. For $L \in CF - REG$ we obtain $L' \notin REG$, hence $K - REG \neq \emptyset$.

Conversely, the language L_2 in the previous section is not in K (we have $L_2^* \cap a^+b^+ = L_2$), hence $L_2 \in CF - K$. □

Corollary 3.8 *K is incomparable with LIN.*

Proof. For $L \in CF - LIN, L' \notin LIN$, but $L_2 \in LIN - K$. □

Theorem 3.9 *The inclusions $K_n \subset K_{n+1}$ are proper for all $n \geq 1$.*

Proof. (1) $n = 1$.

The language

$$L_{a,b} = \{x \in \{a,b\}^* \mid |x|_a \neq |x|_b\}$$

is not regular (its complement, $\{x \in \{a,b\}^* \mid |x|_a = |x|_b\}$, is clearly non-regular), hence it is not in $K_1 = REG$.

However,

$$L_{a,b} \cup L_{a,b}L_{a,b} = \{a,b\}^+.$$

The inclusion $\subseteq$ is obvious. Conversely, if $x \in \{a,b\}^+$, $|x|_a \neq |x|_b$, then $x \in L_{a,b}$. If $|x|_a = |x|_b$, then either $x = ax'$, $|x'|_a < |x'|_b$ or $x = bx'$, $|x'|_a > |x'|_b$. In both cases $x' \in L_{a,b}$, and $a, b \in L_{a,b}$, therefore $x \in L_{a,b}L_{a,b}$.

On the other hand, $L_{a,b} \in CF$. Indeed, consider the context-free grammar

$$G = (\{S, A, B\}, \{a,b\}, S, P),$$

with P containing the following rules:

$$S \to AaA, S \to BbB,$$

$$A \to AA, A \to a, A \to \lambda, A \to aAb, A \to bAa,$$

$$B \to BB, B \to b, B \to \lambda, B \to aBb, B \to bBa.$$

Clearly, starting by $S \to AaA$ we generate strings x with $|x|_a > |x|_b$ and starting by $S \to BbB$ we obtain strings x with $|x|_a < |x|_b$ (from A one generates all the strings x with $|x|_a \geq |x|_b$ and from B one generates all the strings x with $|x|_a \leq |x|_b$).

(2) $n \geq 2$.

Consider the language

$$L_n = L_{a,b} \cup L_{a,b}\{c\}L_{a,b} \cup M_n,$$

for

$$M_n = \{x \in \{a,b,c\}^* \mid |x|_c \geq n\}.$$

Clearly, $L_n \in CF$, but

$$L_n \cap \{a,b\}^* = L_{a,b},$$

hence $L_n \notin REG$. In fact, for all $k, 1 \leq k \leq n$, we have

$$\bigcup_{i=1}^{k} L_n \cap \{x \in \{a,b,c\}^* \mid |x|_c = k-1\} =$$

$$= \{x_1 c x_2 c \ldots c x_k x_{k+1} \mid x_i \in \{a,b\}^+, 1 \leq i \leq k+1,$$
$$|x_j| \geq 2, 2 \leq j \leq k, \text{ and } x_1 \in L_{a,b}, \text{ or } x_{k+1} \in L_{a,b}\}.$$

Denote this language by H. Indeed, $k - 1 < n$, hence $H \cap M_n^* = \emptyset$; it follows that

$$H \subseteq \bigcup (L_{a,b}\{c\}L_{a,b})^i L_{a,b}(L_{a,b}\{c\}L_{a,b})^j,$$

the union being taken for all $i, j \geq 0$ with $i + j = k - 1$.

The language H is not regular: $z_i = a^i caacaa \ldots aaca^i \in H$ for all $i \geq 1$, but every two strings z_i, z_j with $i \neq j$ are not congruent (the context (b^i, b^i) accepts only z_j).

However,

$$\bigcup_{i=1}^{n+1} L_n^i = \{a,b\}^+ \cup M_n \cup$$
$$\cup \{x_1 c x_2 c \ldots x_r c x_{r+1} \mid 1 \leq r \leq n-1, x_i \in \{a,b\}^+, 1 \leq i \leq r+1, |x_j| \geq 2, 2 \leq j \leq n\},$$

hence this language is regular.

The inclusion $\subseteq$ is obvious (note that $M_n^+ = M_n$). Conversely, $M_n \subseteq L_n$, $\{a,b\}^+ = L_{a,b} \cup L_{a,b} L_{a,b}$, and $x_1 c x_2 c \ldots x_r c x_{r+1} \in L_{a,b}(L_{a,b}\{c\}L_{a,b})^r L_{a,b}$ for all $1 \leq r \leq n-1$, $x_i \in \{a,b\}^+$, $1 \leq i \leq r+1$, $|x_j| \geq 2, 2 \leq j \leq r$. (The details are the same as in the first part of the proof.)

In conclusion, $L_n \in K_{n+1} - K_n$ and the proof is complete. □

Theorem 3.10 *$K_n \subset K$ for all $n \geq 1$.*

Proof. The language

$$L = \{a^n b^n \mid n \geq 1\} \cup \{a,b\}$$

is in K but $L \notin K_n$ for $n \geq 1$. Indeed, suppose that $\bigcup_{i=1}^m L^i$ is regular for some m. We have

$$\bigcup_{i=1}^{m} L^i \cap a^* b^* = \{x \in a^* b^* \mid -m \leq |x|_a - |x|_b \leq m\},$$

and this is not a regular language, a contradiction. □

The family CL is quite comprehensive and, in fact, the condition $R \in REG$ in its definition can be removed:

Theorem 3.11 *Assume that $L_1 \neq \emptyset$ and L_2 are arbitrary languages over the alphabet V such that $L_1 L_2 \in REG$. Then also $V^* L_2 \in REG$.*

Proof. Let $x \in L_1$ be a string such that the conditions

$$y \in L_1, \ |y| < |x|,$$

are satisfied for no string y. Since $L_1 L_2$ is regular, so is the left derivative

$$L_0 = d_x^l(L_1 L_2)$$

and, hence, also $V^* L_0$ is regular. Since x is shortest in L_1, we have also

$$L_0 = (d_x^l(L_1)) L_2.$$

Hence,

$$V^* L_0 = (V^* d_x^l(L_1)) L_2 \subseteq V^* L_2.$$

But $L_2 \subseteq L_0$ because $\lambda \in d_x^l(L_1)$. Consequently, $V^* L_2 \subseteq V^* L_0$, which implies that $V^* L_0 = V^* L_2$. Since $V^* L_0$ is regular, so is $V^* L_2$. □

Using right derivatives, it can be shown similarly that if $L_1L_2 \in REG$ and $L_2 \neq \emptyset$, then $L_1V^* \in REG$.

Remark 1. The proof is effective if one of the shortest strings in L_1 can be effectively found. This is the case when, for instance, L_1 is a context-free language.

Corollary 3.12 *$K \subset CL$, strict inclusion.*

Proof. Take $L \subseteq V^*, L \in K$. Therefore $L^* \in REG$. This implies $L^+ = L^* - \{\lambda\} \in REG$, too. Moreover, $L^+ = L^*L$.

According to the previous theorem, $L^*L \in REG$ implies $V^*L \in REG$, hence $L \in CL$ and we have obtained the inclusion $K \subseteq CL$.

This inclusion is proper. For instance, the language L_1 considered in Section 2 is in $CL - K$. Indeed, $L_1^* \cap a^*b^* = L_1$, which is not regular, hence L_1^* is not regular. □

Corollary 3.13 *A context-free language $L \subseteq V^*$ is in CL if and only if $V^*L \in REG$.*

This corollary is useful in showing that languages are not in CL, for instance, in the proof of Theorem 8.

Remark 2. The generality of this result (L_1, L_2 are arbitrary languages) can be compared with the known result (see [5], page 50) that the left quotient of a regular language by an arbitrary language is a regular language, as well as with Lemma 3.1 in [6], which states that also deleting from the strings of a regular language substrings which belong to an arbitrary language, we still obtain a regular language. The previous theorem is in some sense a dual to these results.

A sort of converse of Theorem 5 is natural to be looked for, namely given L_1L_2 regular, it is expected that for any $x \in L_1$, also $(L_1 - \{x\})L_2$ is regular. However, this is not true.

Theorem 3.14 *There are $L_1, L_2 \subseteq \{a,b\}^*, L_1$ linear, L_2 regular, and $x \in L_1$, such that L_1L_2 is regular, but $(L_1 - \{x\})L_2$ is not regular.*

Proof. Consider the language

$$L_1 = \{a^iba^j \mid 1 \leq i < j\} \cup \{a\}.$$

It is clearly linear and

$$\begin{aligned} L_1^* = \{&a^{i_1}ba^{i_2}b \dots a^{i_k}ba^{i_{k+1}} \mid k \geq 1, i_1 \geq 1, \\ &i_s \geq 3, 1 \leq s \leq k, i_{k+1} \geq 2\} \cup a^*. \end{aligned}$$

Consequently, $L_1^* \in REG$. We take $L_2 = L_1^*$. Obviously, $L_1L_2 = L_1^+$ is regular, too. However,

$$(L_1 - \{a\})L_2 \cap a^*ba^* = \{a^iba^j \mid 1 \leq i < j\},$$

which is not a regular language, hence $(L_1 - \{a\})L_2$ is not regular. □

The next theorem will give a characterization of languages in the family K. With this aim, the notion of *root* of a language in the sense of [1] is used (see also [8], pages 126 - 127).

Given a language $L \subseteq V^*$, we denote by $root(L)$ the smallest language $L_0 \subseteq L$ such that $L_0^* = L^*$; it is proved in [1] that such a language exists and it is unique.

Theorem 3.15 *A language $L \in CF$ is in K if and only if there is a regular language $L_0 \subseteq L$ such that $L \subseteq L_0^*$.*

Proof. The *if* part is obvious ($L_0 \subseteq L \subseteq L_0^*$, hence $L^* = L_0^* \in REG$).

Conversely, we have $root(L) = root(L^*)$. For all regular language, M, $root(M)$ is regular, too [1]. Therefore, for $L \in K$, $root(L^*) \in REG$. Thus, we can take $L_0 = root(L) = root(L^*)$, and all conditions in the theorem are satisfied. □

4 Closure and decidability properties

The families $CL, K, K_n, n \geq 2$, have rather poor closure properties.

Theorem 4.1 *The family CL is closed under morphisms and $Pref, Suf, Sub$, but it is not closed under union, concatenation, Kleene +, intersection by regular sets, inverse morphisms and mirror image.*

Proof.

Morphisms. If $L \in CL, L \subseteq V^*$ and $h : V^* \longrightarrow U^*$, then let $R \in REG$ be such that $RL \in REG$. As $h(RL) = h(R)h(L)$, we have $h(RL) \in REG$, hence $h(L) \in CL$.

Pref, Suf, Sub. As a consequence of Lemma 3 (i), if by an operation α, from a language L we obtain $\alpha(L)$ containing the empty string, then $\alpha(L) \in CL$. This is the case with $Pref, Sub, Suf$.

Union. Consider the languages

$$L_1 = \{a^n b^m \mid 0 \leq n \leq m\},$$

$$L_2 = \{c^n d^m \mid 0 \leq n \leq m\},$$

which are both in CL (take $R_1 = a^*, R_2 = c^*$). Since $\{a, b, c, d\}^*(L_1 \cup L_2)$ is not regular, we conclude by Corollary 2 of Theorem 5 that $L_1 \cup L_2 \notin CL$.

Concatenation. The languages

$$L_1 = \{b\},$$
$$L_2 = \{a^n b^m \mid 0 \leq n \leq m\},$$

are in CL, but $L_1 L_2$ is not in CL, again by Corollary 2 of Theorem 5.

Kleene +. For the previous language L_2 we have $L_2^* \notin CL$ (indeed, $L_2^* \cap a^+ b^+ = L_2$).

Intersection by regular sets. As we have seen, D, the Dyck language over $\{a, b\}$, is in CL, but

$$D \cap a^+ b^+ = \{a^n b^n \mid n \geq 1\},$$

which is not in CL.

Inverse morphisms. Take the language

$$L = \{(baa)^n (ab)^m \mid 0 \leq n \leq m\}.$$

It belongs to CL. Consider also the morphism

$$h : \{a, b, c, d, e, f\}^* \longrightarrow \{a, b\}^*$$

defined by

$$h(a) = baa, h(b) = ab, h(c) = b, h(d) = aab, h(e) = aaa, h(f) = ba.$$

We obtain

$$\begin{aligned} h^{-1}(L) &= \{a^n b^m \mid 0 \leq n \leq m\} \cup \\ &\cup \{a^r c d^n e f^m c b^p \mid r, p \geq 0, 0 \leq r + n \leq m + p\} \cup \\ &\cup \{a^r f b d^n e f^m c b^p \mid r, p \geq 0, 0 \leq r + n + 1 \leq m + p\}. \end{aligned}$$

Again Corollary 2 of Theorem 5 shows that $h^{-1}(L) \notin CL$.

Mirror image. The language $\{a^n b^m \mid 0 \leq n \leq m\}$ is in CL, but its mirror image is not. □

Theorem 4.2 *The family K is closed under union, Kleene $*$ and morphisms, but it is not closed under concatenation, intersection by regular sets and inverse morphisms.*

Proof. The positive results follow from the next equalities:

$(L_1 \cup L_2)^* = (L_1^* \cup L_2^*)^*$ (union),
$(L^*)^* = L^*$ (Kleene closure),
$(h(L))^* = h(L^*)$ (morphisms).

Concatenation. Take the languages

$$L_1 = \{a^n b^n \mid n \geq 1\} \cup \{a, b\},$$
$$L_2 = \{c\},$$

both in K. However, $L_1 L_2 \notin K$, because

$$(L_1 L_2)^* \cap a^+ b^+ c = \{a^n b^n c \mid n \geq 1\},$$

a non-regular language.

Intersection by regular sets. For L_1 as above we have

$$L_1 \cap a^+ b^+ = \{a^n b^n \mid n \geq 1\},$$

which is not in K.

Inverse morphisms. Consider the language

$$L = \{a^{2n} b^{2n} \mid n \geq 1\} \cup \{a, b\},$$

which is in K, and the morphism $h : \{a, b\}^* \longrightarrow \{a, b\}^*$ defined by

$$h(a) = aa, h(b) = bb.$$

We have

$$h^{-1}(L) = \{a^n b^n \mid n \geq 1\},$$

which we have seen is not in K □.

Theorem 4.3 *The families $K_n, n \geq 2$, are closed under morphisms and Kleene $*$, but they are not closed under union, concatenation, intersection by regular sets and inverse morphisms.*

Proof.
Morphisms. Use the equality $h(\bigcup_{i=1}^{m} L^i) = \bigcup_{i=1}^{m} h(L^i), m \geq 1$.
Kleene $*$. Follows from the inclusion $K_n \subseteq K, n \geq 1$.
Union. Take

$$L_1 = \{a^s b a^t \mid s \neq t, s, t \geq 1\} \cup a^*,$$
$$L_2 = \{b^2\}.$$

We have

$$\begin{aligned} L_1 \cup L_1 L_1 \;=\; & \{a^s b a^t \mid s, t \geq 1\} \cup a^* \cup \\ & \cup \{a^s b a^t b a^r \mid s, r \geq 1, t \geq 2, \\ & \quad (s,t,r) \notin \{(1,2,1),(1,2,2),(2,2,1),(1,3,1),(2,3,2)\}\}, \end{aligned}$$

hence $L_1 \in K_2$; clearly, $L_2 \in K_1$. However, $L_1 \cup L_2 \notin K_n$, for all given n. Indeed, assume

$$L = \bigcup_{i=1}^{m} (L_1 \cup L_2)^i \in REG,$$

for some m. If $m = 2k, k \geq 1$, then we have

$$L \cap (a^* b a^* b^2)^k = \{a^s b a^t b^2 \mid s \neq t, s, t \geq 1\}^k,$$

which is not regular. If $m = 2k+1, k \geq 1$, then

$$L \cap (a^* b a^* b^2)^k a^* b a^* = \{a^s b a^t b^2 \mid s \neq t, s, t \geq 1\}^k \{a^s b a^t \mid s \neq t, s, t \geq 1\},$$

which is non-regular, too.
Concatenation. For the above languages L_1, L_2, take $L_1 L_2$, then follow an argument similar as for union.
Intersection with regular sets. Take again L_1 and intersect it by $a^* b a^*$. We have

$$(\bigcup_{i=1}^{m} (L_1 \cap a^* b a^*)^i) \cap a^* b a^* = \{a^s b a^t \mid s \neq t, s, t \geq 1\},$$

which is not regular.
Inverse morphisms. Consider the language

$$L = \{(ab)^s b (ab)^t \mid s \neq t, s, t \geq 1\} \cup (ab)^*$$

and the morphism $h : \{a, b, c, d\}^* \longrightarrow \{a, b\}^*$, defined by

$$h(a) = a, h(b) = ba, h(c) = bba, h(d) = b.$$

As for L_1, we have $L \in K_2$. Clearly,

$$h^{-1}(L) = \{ab^{s-1} c b^{t-1} d \mid s \neq t, s, t \geq 1\} \cup \{ab^r d \mid r \geq 0\},$$

hence, for all $m \geq 1$,

$$(\bigcup_{i=1}^{m} h^{-1}(L)^i) \cap ab^* c b^* d = \{ab^{s-1} c b^{t-1} d \mid s \neq t, s, t \geq 1\},$$

which is not regular, hence $h^{-1}(L) \notin K_n$, for $n \geq 2$. □

Corollary 4.4 *No family $CL, CR, K, K_n, n \geq 2$, is an AFL or an anti-AFL.*

The following undecidability result is somewhat expected.

Corollary 4.5 *It is undecidable whether or not an arbitrarily given context-free language over an alphabet with at least two symbols is in CL (in K or in $K_n, n \geq 1$).*

Proof. Take $L \subseteq \{a,b\}^*$ arbitrary in CF and the morphism $h : \{a,b\}^* \longrightarrow \{a,b\}^*$, defined by

$$h(a) = bab, h(b) = baab.$$

Since $L = h^{-1}(h(L))$, the language $h(L)$ is regular iff L is regular.

We construct the language

$$L' = \{ba^3b\}h(L).$$

Then, $L' \in CL$ (and $L' \in K, L' \in K_n, n \geq 1$, respectively) iff L is regular (which is undecidable).

Indeed,

1. $\{a,b\}^*L' \in REG$ if and only if $L \in REG$.
   - (if) Obvious.
   - (only if) We have
   $$L = h^{-1}(d^l_{ba^3b}(Suf(\{a,b\}^*L') \cap \{ba^3b\}\{a,b\}^*)).$$
2. $\bigcup_{i=1}^n L'^i \in REG$ if and only if $L \in REG$, for all $n = 2, 3, \ldots, \infty$.
   - (if) Obvious.
   - (only if) We have $L = h^{-1}(d^l_{ba^3b}(\bigcup_{i=1}^n L'^i \cap \{ba^3b\}\{a,b\}^*)), n \geq 2$. □

References

[1] J. A. Brzozowski, Roots of star events, *Journal of the ACM*, 14 (1967), 466 - 477.

[2] W. Bucher, A. Ehrenfeucht, D. Haussler, On total regulators generated by derivation relations, *Theor. Computer Sci.*, 40 (1985), 131 - 148.

[3] J. Dassow, Gh. Paun, On the degree of non-regularity of context-free languages, *Intern. J. Computer Math.*, 36 (1990), 13 - 29.

[4] A. Ehrenfeucht, D. Haussler, G. Rozenberg, On regularity of context-free languages, *Theor. Computer Sci.*, 27 (1983), 311 - 332.

[5] M. A. Harrison, *Introduction to Formal Language Theory*, Addison Wesley, Reading, Mass., 1978.

[6] L. Kari, *On Insertion and Deletion in Formal Language Theory*, Ph.D. Thesis, Univ. of Turku, Dept. of Mathematics, 1991.

[7] E. Makinen, Two complexity measures for context-free languages, *Intern. J. Computer Math.*, 26 (1988), 29 - 34.

[8] A. Salomaa, *Theory of Automata*, Pergamon Press, New York, 1969.

[9] A. Salomaa, *Formal Languages*, Academic Press, New York, London, 1973.

Received December 3, 1992

Acta Cybernetica, Vol. 10, Nr. 4, Szeged, 1992

The Boolean Closure of DR-Recognizable Tree Languages

E. Jurvanen*

Abstract

The family *DRec* of tree languages recognized by deterministic root-to-frontier (top-down) tree automata is not closed under unions or complements. Hence, it is not a variety of tree languages in the sense of Steinby. However, we show that the Boolean closure of *DRec* is a variety which is properly included in the variety *Rec* of all recognizable tree languages. This Boolean closure is also compared with some other tree language varieties.

1 Introduction

Finite tree recognizers are divided into four types according to whether they are deterministic or not, and whether they read trees from root to frontier or from frontier to root. The nondeterministic tree automata and the deterministic frontier-to-root tree automata recognize the same class of tree languages. This is the class of recognizable tree languages which is here denoted by *Rec*. However, the deterministic root-to-frontier tree automata recognize a proper subclass of *Rec* called here *DRec*. These tree automata types were defined and the connections between the languages they recognize were established in the late sixties by Thatcher and Wright [TW68], Rabin [Rab69], Doner [Don70], Magidor and Moran [MM69].

The class *DRec* has been studied relatively little. Courcelle [Cou78a,Cou78b] and Virágh [Vir80] gave a characterization using a path closure operator. Gécseg and Steinby [GS78] presented an algorithm for minimizing deterministic root-to-frontier tree automata.

In this paper we study the Boolean closure of *DRec* denoted here by $\mathcal{B}(DRec)$. It is shown to form a variety in the sense of Steinby [Ste79,Ste92]. Since also *Rec* is a variety, the next question is, whether variety $\mathcal{B}(DRec)$ is properly included in variety *Rec*. In connection with his studies of logic characterizations of tree language families, Thomas [Tho84] answered this question positively; $\mathcal{B}(DRec)$ is a proper subclass of the chain definable tree languages which form a proper subclass of *Rec*. In this work we also prove the proper inclusion of $\mathcal{B}(DRec)$ in *Rec*, but directly using only the pidgeon hole principle. After that $\mathcal{B}(DRec)$ is compared with respect to the inclusion relation with the varieties *Nil, D, RD, GD* and *Loc*, where *Nil* is the Boolean closure of the family of finite tree languages, and the others consist of the definite, the reverse definite, the generalized definite and the local tree languages, respectively. Some of the definitions of these tree families were

*Department of Mathematics, University of Turku, SF-20500 Turku, Finland, E-mail: jurvanen@utu.fi

given by Heuter [Heu88,Heu89a,Heu89b] and they were shown to be varieties by Steinby [Ste92].

The notation is mostly from [GS84].

2 Preliminaries

For a set A, we denote by pA the power set of A, that is the set of all subsets of A, and by $|A|$ the cardinality of A. If $A \subseteq B$, but $A \neq B$, then we write $A \subset B$.

Let N be the set of natural numbers, $N = \{0, 1, \ldots\}$. A *ranked alphabet* Σ is a finite set of operation symbols each of which has been assigned a unique rank from N. For $m \in N$, the set of m-ary operation symbols of Σ form a set denoted by Σ_m. Thus $\Sigma = \bigcup_{m \in N} \Sigma_m$. Two special cases are the *trivial* ranked alphabets, for which $\Sigma = \Sigma_0$, and the *unary* ranked alphabets satisfying $\Sigma = \Sigma_0 \cup \Sigma_1$.

In a *Σ-algebra* $\mathcal{A} = (A, \Sigma)$, A is a nonempty set, Σ is a set of operation symbols and every operation symbol $\sigma \in \Sigma_m$, where $m \geq 1$, is interpreted as a mapping

$$\sigma^{\mathcal{A}} : A^m \longrightarrow A,$$

and every nullary symbol $\sigma \in \Sigma_0$ is interpreted as an element $\sigma^{\mathcal{A}}$ of A. If $\mathcal{A} = (A, \Sigma)$ and $\mathcal{B} = (B, \Sigma)$ are two Σ-algebras, then a *homomorphism* from $\mathcal{A}$ to $\mathcal{B}$ is a mapping $\phi : A \longrightarrow B$ such that

$$\sigma^{\mathcal{A}}(a_1, \ldots, a_m)\phi = \sigma^{\mathcal{B}}(a_1\phi, \ldots, a_m\phi)$$

holds for all $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in A$. In particular, if $\sigma \in \Sigma_0$ and ϕ is a homomorphism, then $\sigma^{\mathcal{A}}\phi = \sigma^{\mathcal{B}}$. An equivalence relation θ on A is a *congruence* of $\mathcal{A}$, if for all $m \geq 0, \sigma \in \Sigma_m$ and $a_1, \ldots, a_m, b_1, \ldots, b_m \in A$,

$$a_1\theta b_1, \ldots, a_m\theta b_m \quad \text{implies} \quad \sigma^{\mathcal{A}}(a_1, \ldots, a_m)\theta\sigma^{\mathcal{A}}(b_1, \ldots, b_m).$$

An equivalence class of a congruence is called a *congruence class* and the congruence class of $a \in A$ is denoted by $a\theta$. A congruence of $\mathcal{A}$ is said to *saturate* a subset $L \subseteq A$, if $L = L\theta$. This means that L is the union of some congruence classes of θ. If a congruence has finitely many congruence classes, then the congruence is *finite*.

Let X be an alphabet, that is a finite set of letters, such that $\Sigma \cap X = \emptyset$. We assume also that $X \cup \Sigma_0 \neq \emptyset$. The set of all *$\Sigma X$-trees* is the smallest set containing every $x \in X, \sigma \in \Sigma_0$ and $\sigma(t_1, \ldots, t_m)$, where $m \geq 1, \sigma \in \Sigma_m$ and $t_1, \ldots, t_m$ are ΣX-trees. A set of ΣX-trees is called a *ΣX-forest* or a *ΣX-tree language*. The set of all ΣX-trees is denoted by $F_\Sigma(X)$. The *complement* of a ΣX- forest T is $T^C = F_\Sigma(X) \setminus T$. The height, root, subtrees and leaves of a tree t are denoted by $\mathrm{hg}(t)$, $\mathrm{root}(t)$, $\mathrm{sub}(t)$ and $\mathrm{leaf}(t)$ respectively. As usual, if $t \in X \cup \Sigma_0$, then $\mathrm{hg}(t) = 0$, $\mathrm{root}(t) = t$ and $\mathrm{sub}(t) = \{t\}$. For $t = \sigma(t_1, \ldots, t_m)$, where $m > 0$, $\mathrm{hg}(t) = 1 + \max_{1 \leq i \leq m} \mathrm{hg}(t_i)$, $\mathrm{root}(t) = \sigma$ and $\mathrm{sub}(t) = \{t\} \cup \bigcup_{1 \leq i \leq m} \mathrm{sub}(t_i)$. The leaves of any tree are its subtrees of height 0.

Let ξ be a letter not in $X \cup \Sigma$. A tree $p \in F_\Sigma(X \cup \{\xi\})$ is a *special tree*, if ξ occurs in it exactly once. The set of all special trees is denoted by $Sp_\Sigma(X)$. The *product* of a special tree $p \in F_\Sigma(X \cup \{\xi\})$ and a tree $t \in F_\Sigma(X)$ is a tree $t \cdot_\xi p \in F_\Sigma(X)$, which is formed from p by substituting t for its leaf ξ. When $p \in Sp_\Sigma(X)$ and $T \subseteq F_\Sigma(X)$, the *p-translation* of T is

$$p(T) = \{t \cdot_\xi p \mid t \in T\}$$

and the *inverse p-translation* of T is

$$p^{-1}(T) = \{t \in F_\Sigma(X) \mid t \cdot_\xi p \in T\}.$$

The ΣX-trees form a Σ-algebra $\mathcal{F}_\Sigma(X) = (F_\Sigma(X), \Sigma)$ with operations from Σ defined as

$$\sigma^{\mathcal{F}_\Sigma(X)}(t_1, \ldots, t_m) = \sigma(t_1, \ldots, t_m),$$

where $m \geq 0, t_1, \ldots, t_m \in F_\Sigma(X)$ and $\sigma \in \Sigma_m$. This Σ-algebra is called the *ΣX-term algebra.*

A set of trees which can be recognized by a frontier-to-root or a nondeterministic root-to-frontier recognizer [GS84] is called *recognizable*. The set of all recognizable ΣX-tree languages we denote by $Rec(\Sigma, X)$.

A *deterministic root-to-frontier Σ-algebra* (a DR Σ-algebra) is a pair $\mathcal{A} = (A, \Sigma)$, where A is a nonempty set and every operation symbol $\sigma \in \Sigma_m$ with $m > 0$ is interpreted as a mapping

$$\sigma^{\mathcal{A}} : A \to A^m.$$

If $\sigma \in \Sigma_0$, then it defines a singleton $\sigma^{\mathcal{A}}$ in A. An algebra $\mathcal{A} = (A, \Sigma)$ is called *finite,* if the set A is finite.

Let X be an alphabet. A *deterministic root-to-frontier ΣX-recognizer* (a DR ΣX-recognizer) is a triple $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$, where

(1) $\mathcal{A}$ is a finite DR Σ-algebra $\mathcal{A} = (A, \Sigma)$,
(2) $a_0 \in A$ is the *initial state* and
(3) $\alpha : X \to pA$ is the *final assignment.*

The recognizer is also denoted by $\mathbf{A} = (A, \Sigma, X, a_0, \alpha)$. The elements of the set A are called the *states* of the recognizer.

Next we define the language which a DR ΣX-recognizer $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ accepts. We need the mapping $\tilde{\alpha} : F_\Sigma(X) \to pA$, which is defined as follows:

(1) If $x \in X$, then $x\tilde{\alpha} = x\alpha$.
(2) If $\sigma \in \Sigma_0$, then $\sigma\tilde{\alpha} = \{\sigma^{\mathcal{A}}\}$.
(3) If $t = \sigma(t_1, \ldots, t_m)$, where $m \geq 1$, then

$$t\tilde{\alpha} = \{a \in A | \sigma^{\mathcal{A}}(a) \in (t_1\tilde{\alpha} \times \ldots \times t_m\tilde{\alpha})\}.$$

Now the forest recognized by $\mathbf{A}$ is the set

$$T(\mathbf{A}) = \{t \in F_\Sigma(X) | a_0 \in t\tilde{\alpha}\}.$$

A forest that can be recognized by a DR ΣX-recognizer is called *DR-recognizable* or simply a *DRec-language.* The set of all DR-recognizable ΣX-tree languages is $DRec(\Sigma, X)$.

Because a deterministic recognizer can always be regarded as nondeterministic, a DR-recognizable language is also recognizable. Thus $DRec(\Sigma, X) \subseteq Rec(\Sigma, X)$.

Lemma 2.1 *If $\Sigma \neq \Sigma_0 \cup \Sigma_1$, then $DRec(\Sigma, X)$ is properly included in $Rec(\Sigma, X)$.*

Proof. We generalize a tree language originally due to Magidor and Moran [MM69] and simplified by Thatcher [Tha73]. Let $x \in X \cup \Sigma_0$ and $\sigma \in \Sigma_m$ for $m \geq 2$. Then the forest $\{\sigma(\sigma(x, \ldots, x), x, \ldots, x), \sigma(x, \ldots, x, \sigma(x, \ldots, x))\}$ belongs to $Rec(\Sigma, X)$, but it is not DR-recognizable. □

Next we define the paths of a tree and the path closure of a forest. Using the path closure concept we can distinguish *DRec*-languages among *Rec*-languages. Then we can easily see that any intersection of finitely many *DRec*-languages is also a *DRec*-language, but that the Boolean closure of *DRec*-languages properly contains the *DRec*-languages themselves.

Let Σ be a ranked alphabet. For every operation symbol $\sigma \in \Sigma_m$ $(m > 0)$, we define a set of new unary operations $\Gamma(\sigma) = \{\sigma_1, \ldots, \sigma_m\}$ so that if $\sigma \neq \tau$, then $\Gamma(\sigma) \cap \Gamma(\tau) = \emptyset$. Then we form a new alphabet $\Gamma = \Gamma_0 \cup \Gamma_1$, where

(1) $\Gamma_0 = \Sigma_0$ and
(2) $\Gamma_1 = \bigcup\{\Gamma(\sigma) | \sigma \in \Sigma_m, m \geq 1\}$.

The *paths* of a tree $t \in F_\Sigma(X)$ form the set $\delta(t) \subseteq F_\Gamma(X)$ defined as follows:

(1) For $x \in X$, let $\delta(x) = \{x\}$.
(2) For $\sigma \in \Sigma_0$, let $\delta(\sigma) = \{\sigma\}$.
(3) For $t = \sigma(t_1, \ldots, t_m)$, let

$$\delta(t) = \bigcup_{i=1}^{m} \sigma_i(\delta(t_i)).$$

Now the set of the paths of a forest $T \subseteq F_\Sigma(X)$ is

$$\delta(T) = \bigcup\{\delta(t) | t \in T\}$$

and its *path closure* $\Delta(T)$ is the forest

$$\Delta(T) = \{t \in F_\Sigma(X) | \delta(t) \subseteq \delta(T)\}.$$

For example, the path set of the tree $t = \sigma(z, \sigma(\omega(y), x))$ contains the elements $\sigma_1(z), \sigma_2(\sigma_1(w_1(y)))$ and $\sigma_2(\sigma_2(x))$, and the path closure of the forest $T = \{\sigma(x, y), \sigma(y, x)\}$ is $\Delta(T) = T \cup \{\sigma(x, x), \sigma(y, y)\}$.

Some of the properties of the path closure are noted in the following lemma.

Lemma 2.2 [Vir80]. *If* $T, T_1, T_2 \subseteq F_\Sigma(X)$, *then*

(1) $T \subseteq \Delta(T)$,
(2) $\Delta(T) = \Delta(\Delta(T))$ *and*
(3) $T_1 \subseteq T_2$ *implies* $\Delta(T_1) \subseteq \Delta(T_2)$. □

Theorem 2.3 [Cou78b, Vir80]. *Let* $T \in Rec(\Sigma, X)$ *and* $\Sigma_0 = \emptyset$. *Then*

$$T \in DRec(\Sigma, X) \quad \text{iff} \quad \Delta(T) = T.$$

□

Corollary 2.4 *Let* $S, T \subseteq F_\Sigma(X)$. *Then*

$$S, T \in DRec(\Sigma, X) \quad \text{implies} \quad S \cap T \in DRec(\Sigma, X).$$

Proof. We present a short proof in the case $\Sigma_0 = \emptyset$. For a general ranked alphabet, a product of two DR-recognizers can be constructed that accepts the intersection. If $S, T \in DRec(\Sigma, X)$, then $S \cap T \in Rec(\Sigma, X)$, because $Rec(\Sigma, X)$ is closed under Boolean operations.

Assume $t \in \Delta(S \cap T)$. Then $\delta(t) \subseteq \delta(S \cap T) \subseteq \delta(S) \cap \delta(T)$. Now $t \in \Delta(S) \cap \Delta(T) = S \cap T$. Thus $S \cap T \in DRec(\Sigma, X)$. □

Let $\mathcal{F}$ be a family of subsets of a set U. The *Boolean closure* $\mathcal{B}(\mathcal{F})$ of $\mathcal{F}$ is the smallest set $\mathcal{G}$ of subsets of U which contains $\mathcal{F}$ such that $X, Y \in \mathcal{G}$ implies $X \cap Y, X \cup Y$ and $U \backslash X \in \mathcal{G}$. We denote the complement $U \backslash X$ of X by X^C. The following theorem can be found, for instance, in [Sik64].

Theorem 2.5 *Let $\mathcal{F} \subseteq pU$ be a family of subsets of a given set U. A subset T of U is in the Boolean closure of $\mathcal{F}$ iff there exist $k \geq 1, n_1, \ldots, n_k \geq 1$ and $m_1, \ldots, m_k \geq 0$ such that T can be expressed in the form*

$$\begin{aligned} T \;=\; & (F_{11} \cap F_{12} \cap \cdots \cap F_{1m_1} \cap F^C_{1,m_1+1} \cap F^C_{1,m_1+2} \cap \cdots \cap F^C_{1n_1}) \cup \\ & (F_{21} \cap F_{22} \cap \cdots \cap F_{2m_2} \cap F^C_{2,m_2+1} \cap F^C_{2,m_2+2} \cap \cdots \cap F^C_{2n_2}) \cup \\ & \quad \vdots \\ & (F_{k1} \cap F_{k2} \cap \cdots \cap F_{km_k} \cap F^C_{k,m_k+1} \cap F^C_{k,m_k+2} \cap \cdots \cap F^C_{kn_k}), \end{aligned}$$

where $F_{ij} \in \mathcal{F}$ for every $1 \leq i \leq k$ and every $1 \leq j \leq n_i$. □

Corollary 2.4 and Theorem 2.5 give the following result.

Corollary 2.6 *A set $T \subseteq Rec(\Sigma, X)$ belongs to $\mathcal{B}(DRec(\Sigma, X))$ iff there exist $k \geq 1$ and $n_1, \ldots, n_k \geq 1$ such that T can be presented in the form*

$$\begin{aligned} T \;=\; & (T_{11} \cap T^C_{12} \cap T^C_{13} \cap \cdots \cap T^C_{1n_1}) \cup \\ & (T_{21} \cap T^C_{22} \cap T^C_{23} \cap \cdots \cap T^C_{2n_2}) \cup \\ & \quad \vdots \\ & (T_{k1} \cap T^C_{k2} \cap T^C_{k3} \cap \cdots \cap T^C_{kn_k}), \end{aligned}$$

where for all $1 \leq i \leq k$, $1 \leq j \leq n_i$, the language $T_{ij} \in DRec(\Sigma, X)$. □

Since one-element tree language $\{t\}$ is always DR-recognizable, every finite language belongs to $\mathcal{B}(DRec(\Sigma, X))$. The language $T = \{\sigma(\sigma(x, \ldots, x), x, \ldots, x), \sigma(x, \ldots, x, \sigma(x, \ldots, x))\}$ does not belong to $DRec(\Sigma, X)$ according to the proof of Lemma 2.1, but as finite it is in $\mathcal{B}(DRec(\Sigma, X))$. This observation gives Theorem 2.7.

Theorem 2.7 *If $\Sigma \neq \Sigma_0 \cup \Sigma_1$, then $DRec(\Sigma, X)$ is properly included in $\mathcal{B}(DRec(\Sigma, X))$.* □

3 $\mathcal{B}(DRec)$ is a Variety

Next we show that the Boolean closure of the DR-recognizable languages is a tree language variety. For a family of tree languages to form a variety it is required to be closed under Boolean operations, inverse translations and inverse homomorphisms [Ste79,Ste92].

Let Σ be fixed. If one has defined for every alphabet X a set $\mathcal{V}(X)$ of recognizable ΣX-tree languages, then the family $\mathcal{V} = \{\mathcal{V}(X)\}$ is called a *family of regular Σ-tree languages*. For instance, $Rec = \{Rec(\Sigma, X)\}$ itself, $Triv = \{\{\emptyset, F_\Sigma(X)\}\}$ and $\mathcal{B}(DRec) = \{\mathcal{B}(DRec(\Sigma, X))\}$ are families of regular Σ-tree languages.

Definition 3.1 *Let Σ be fixed. A* variety of Σ-tree languages *is a family of regular Σ-tree languages $\mathcal{V} = \{\mathcal{V}(X)\}$ such that the conditions*

(1) $\emptyset \neq \mathcal{V}(X) \subseteq Rec(\Sigma, X)$,
(2) $T \in \mathcal{V}(X)$ *implies* $F_\Sigma(X) \backslash T \in \mathcal{V}(X)$,
(3) $T, U \in \mathcal{V}(X)$ *implies* $T \cap U \in \mathcal{V}(X)$,
(4) $T \in \mathcal{V}(X), p \in Sp_\Sigma(X)$ *implies* $p^{-1}(T) \in \mathcal{V}(X)$, *and*
(5) *if* $\phi : \mathcal{F}_\Sigma(X) \to \mathcal{F}_\Sigma(Y)$ *is a morphism and* $T \in \mathcal{V}(Y)$, *then* $T\phi^{-1} \in \mathcal{V}(X)$

are satisfied for all alphabets X and Y.

For example, *Triv* and *Rec* are varieties of Σ-tree languages [Ste92]. The family $\mathcal{B}(DRec)$ is closed under Boolean operations by definition. So we need to study the inverse translations and the inverse homomorphisms.

A translation is based on the notion of a special tree. To show that an inverse image of a DR-recognizable language under a translation is again DR-recognizable we also need the concept of a run tree. The idea of a run tree is to associate with every node of a tree the state in which the recognizer has reached that node. Of course, the states associated depend on the initial state at the root. The run tree is defined using the alphabet $X \cup \{\xi\}$ to facilitate handling of special trees as well.

Let $\mathbf{A} = (A, \Sigma, X, a_0, \alpha)$ be a DR ΣX-recognizer and $\xi \notin X \cup \Sigma$. Then the *run tree* of a tree $t \in F_\Sigma(X \cup \{\xi\})$ in state $a \in A$ is $\mathrm{run}(\mathcal{A}, t, a) \in F_{\Sigma \times A}((X \cup \{\xi\}) \times A)$ defined as follows:

(1) If $y \in X \cup \{\xi\}$, then $\mathrm{run}(\mathcal{A}, y, a) = (y, a)$.
(2) If $\sigma \in \Sigma_0$, then $\mathrm{run}(\mathcal{A}, \sigma, a) = (\sigma, a)$.
(3) If $t = \sigma(t_1, \ldots, t_m)$, where $m \geq 1$ and $\sigma^{\mathcal{A}}(a) = (a_1, \ldots, a_m)$, then

$$\mathrm{run}(\mathcal{A}, t, a) = (\sigma, a)(\mathrm{run}(\mathcal{A}, t_1, a_1), \ldots, \mathrm{run}(\mathcal{A}, t_m, a_m)).$$

If the algebra $\mathcal{A}$ is clear from the context, we denote a run tree also by $\mathrm{run}(t, a)$.

With the help of a run tree we get a new way to find out whether a DR-recognizer accepts a tree.

Lemma 3.2 *Let $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ be a DR ΣX-recognizer. Then*

$$t \in T(\mathbf{A}) \text{ iff } a \in l\tilde{\alpha} \text{ for all } (l, a) \in \mathrm{leaf}(\mathrm{run}(\mathcal{A}, t, a_0)).$$

Proof. By tree induction on t one can first prove that $b \in t\tilde{\alpha}$ if and only if $a \in l\tilde{\alpha}$ for all leaves $(l, a) \in \mathrm{leaf}(\mathrm{run}(\mathcal{A}, t, b))$. Choosing then a_0 for b we get the claim. □

Before the main theorem of this subsection we need to study the product of run trees. This is done using the ξ-depth of a special tree.

The *ξ-depth* $\mathrm{dp}(p)$ of a special tree $p \in Sp_\Sigma(X)$ is the length of the path from the root to the ξ-leaf:

(1) If $p = \xi$, then $\mathrm{dp}(p) = 0$.
(2) If $p = \sigma(p_1, \ldots, p_i, \ldots, p_m)$, where $p_i \in Sp_\Sigma(X)$, then $\mathrm{dp}(p) = \mathrm{dp}(p_i) + 1$.

Lemma 3.3 *Let $\mathbf{A} = (A, \Sigma, X, a_0, \alpha)$ be a DR ΣX-recognizer. Let $t \in F_\Sigma(X), p \in Sp_\Sigma(X)$ and let a and b be states of $\mathbf{A}$. If $(\xi, b) \in \mathrm{leaf}(\mathrm{run}(p, a))$, then*

$$\mathrm{leaf}(\mathrm{run}(t \cdot_\xi p, a)) = \mathrm{leaf}(\mathrm{run}(t, b)) \cup \mathrm{leaf}(\mathrm{run}(p, a)) \backslash \{(\xi, b)\}.$$

Proof. By induction on the ξ-depth of the special tree p one can first verify

$$\mathrm{run}(t \cdot_\xi p, a) = \mathrm{run}(t, b) \cdot_{(\xi,b)} \mathrm{run}(p, a),$$

from which the claim follows. □

Theorem 3.4 *Let $p \in Sp_\Sigma(X)$ and $T \subseteq F_\Sigma(X)$. If T is DR-recognizable, then also $p^{-1}(T)$ is DR-recognizable.*

Proof. If $p^{-1}(T) = \emptyset$, then $p^{-1}(T) \in DRec(\Sigma, X)$. Thus we assume that $p^{-1}(T) \neq \emptyset$.

Let $\mathbf{A} = (A, \Sigma, X, a_0, \alpha)$ be a DR ΣX-recognizer that recognizes the forest T. Because $p^{-1}(T) \neq \emptyset$ there exists $t \in p^{-1}(T)$ which means $t \cdot_\xi p \in T = T(\mathbf{A})$. Then by Lemma 3.2 $a \in l\tilde{\alpha}$ for all $(l, a) \in \mathrm{leaf}(\mathrm{run}(\mathcal{A}, t \cdot_\xi p, a_0))$. Since p is a special tree there exists exactly one state $b \in A$ such that $(\xi, b) \in \mathrm{leaf}(\mathrm{run}(\mathcal{A}, p, a_0))$. Now according to Lemma 3.3 we have $a \in l\tilde{\alpha}$ specifically for all $(l, a) \in \mathrm{leaf}(\mathrm{run}(\mathcal{A}, p, a_0)) \backslash \{(\xi, b)\}$.

Form the new recognizer $\mathbf{B} = (A, \Sigma, X, b, \alpha)$ that differs from $\mathbf{A}$ only by its initial state. Of course, also $\mathbf{B}$ is a DR ΣX-recognizer.

Now we show that $T(\mathbf{B}) = p^{-1}(T)$ and so $p^{-1}(T) \in DRec(\Sigma, X)$: for any ΣX-tree t,

$t \in T(\mathbf{B})$
iff $a \in l\tilde{\alpha}$ for all $(l, a) \in \mathrm{leaf}(\mathrm{run}(\mathcal{A}, t, b))$
iff $a \in l\tilde{\alpha}$ for all $(l, a) \in \mathrm{leaf}(\mathrm{run}(\mathcal{A}, t, b)) \cup \mathrm{leaf}(\mathrm{run}(\mathcal{A}, p, a_0)) \backslash \{(\xi, b)\}$
iff $a \in l\tilde{\alpha}$ for all $(l, a) \in \mathrm{leaf}(\mathrm{run}(\mathcal{A}, t \cdot_\xi p, a_0))$
iff $t \cdot_\xi p \in T(\mathbf{A})$
iff $t \in p^{-1}(T)$. □

The inverse image of a DR-recognizable forest under a homomorphism is studied in Theorem 3.5.

Theorem 3.5 *Let $\phi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$ be a homomorphism and let $T \in F_\Sigma(Y)$. If T is DR-recognizable, then also $T\phi^{-1} = \{t | t\phi \in T\} (\subseteq F_\Sigma(X))$ is DR-recognizable.*

Proof. Let $\mathbf{A} = (A, \Sigma, Y, a_0, \alpha)$ be a DR ΣY-recognizer that recognizes the forest T. Form a new recognizer $\mathbf{B} = (A, \Sigma, X, a_0, \beta)$ which differs from $\mathbf{A}$ by its alphabet and its final assignment. The mapping $\beta : X \to pA$ is defined by putting $x\beta = x\phi\tilde{\alpha}$ for all $x \in X$. Also B is a DR ΣX-recognizer.

A proof by tree induction shows that $t\tilde{\beta} = t\phi\tilde{\alpha}$ for all $t \in F_\Sigma(X)$. Hence $t \in T(\mathbf{B})$ if and only if $t\phi \in T(\mathbf{A})$. This means that $T\phi^{-1} = T(\mathbf{B})$ is DR-recognizable. □

According to Theorem 3.4 and Theorem 3.5, every $\mathcal{B}(DRec(\Sigma, X))$ satisfies the conditions of Definition 3.1.

Theorem 3.6 *The family* $\mathcal{B}(DRec) = \{\mathcal{B}(DRec(\Sigma, X))\}$ *is a variety of* Σ*-tree languages.* □

4 $\mathcal{B}(DRec)$ is Properly Included in Rec

Next we show that there is a recognizable tree language that can not be constructed from DR-recognizable languages by finitely many Boolean operations. The proof is based on the pidgeon hole principle and uses Corollary 2.6.

In the beginning of this section we assume that $\Sigma_2 \neq \emptyset$ and that there are at least two variables in X, but later the results are generalized.

A tree $t \in F_\Sigma(X)$ is *balanced*, if all its paths have the same length. Denote the set of all balanced ΣX-trees of height h by $\mathrm{Bal}(h)$.

Let $\sigma \in \Sigma_2$ and $x, y \in X$. A balanced tree $t \in F_{\{\sigma\}}(X)$ is a *left xy-tree*, if $\mathrm{hg}(t) \geq 1$, $\{s \in \mathrm{sub}(t) | \mathrm{hg}(s) \leq 1\} \subseteq \{\sigma(x,y), \sigma(y,x), x, y\}$ and $\sigma(x,y)$ does not appear in t to the right of an occurrence of $\sigma(y,x)$. Thus in a left xy-tree all its subtrees $\sigma(x,y)$ are on the left-hand side and the subtrees $\sigma(y,x)$ are on the right. Denote the set of all left xy-trees of height h by $\mathrm{BLxy}(h)$, where $h \geq 1$. Then $\mathrm{BLxy}(h) \subseteq \mathrm{Bal}(h) \cap F_{\{\sigma\}}(X)$.

The trees in $\mathrm{BLxy}(h)$ differ from each other according to where the leftmost subtree $\sigma(y,x)$ occurs. This also determines how many subtrees $\sigma(x,y)$ it has. We now denote the tree in $\mathrm{BLxy}(h)$ with $n-1$ subtrees $\sigma(x,y)$ by $b(h,n)$, and say that it has the leftmost subtree $\sigma(y,x)$ *at place* n. The tree $b(3,4)$ is displayed in Figure 1 later.

A balanced binary tree of height $h-1$ has 2^{h-1} leaves. When these leaves are then replaced by subtrees $\sigma(x,y)$ and $\sigma(y,x)$, the place for the leftmost subtree $\sigma(y,x)$ can be chosen in $2^{h-1}+1$ ways. So there exist $2^{h-1}+1$ trees in $\mathrm{BLxy}(h)$. Hence

$$\mathrm{BLxy}(h) = \{b(h,n) | n = 1, \ldots, 2^{h-1}+1\}.$$

We also need a mapping $\Omega : \mathrm{BLxy}(h) \to \mathrm{Bal}(h)$ which replaces the leftmost $\sigma(y,x)$ by $\sigma(x,x)$. If a tree has no $\sigma(y,x)$ at all, then Ω leaves the tree unaltered, i.e. $\Omega(b(h,2^{h-1}+1)) = b(h,2^{h-1}+1)$. Note that Ω is an injection.

Lemma 4.1 *Let* $TB = \{b(h,n_1), \ldots, b(h,n_p)\} \subseteq \mathrm{BLxy}(h)$, *where* $n_1 < n_2 < \cdots < n_p$. *Then*

$$\Omega(TB \backslash \{b(h,n_p)\}) \subseteq \Delta(TB).$$

Proof. Consider the tree $\Omega(b(h,n_i))$, where $1 \leq i < p$. At place n_i it has a subtree $\sigma(x,x)$, and this is the only place where it differs from the original tree $b(h,n_i)$, which has a subtree $\sigma(y,x)$ at place n_i. The tree $b(h,n_p)$ has a subtree $\sigma(x,y)$ at place n_i. Thus $\Omega(b(h,n_i)) \in \Delta(\{b(h,n_i), b(h,n_p)\}) \subseteq \Delta(TB)$. □

Lemma 4.2 *Let* $t_1 = b(h,n_1)$ *and* $t_2 = b(h,n_2)$, *where* $n_1 < n_2$. *Then*

$$n_1 < n \leq n_2 \quad \text{implies} \quad b(h,n) \in \Delta(\{\Omega(t_1), \Omega(t_2)\}).$$

Proof. Consider a tree $b(h,n)$, where $n_1 < n \leq n_2$. Left to the place n it has only subtrees $\sigma(x,y)$ just like the tree $\Omega(t_2)$. At place n and right to it the tree $b(h,n)$ has only subtrees $\sigma(y,x)$ just like the tree $\Omega(t_1)$. □

Lemma 4.3 *Let $\sigma \in \Sigma_2, x, y \in X$ and $T \subseteq F_\Sigma(X)$. If no tree in T has a subtree $\sigma(x,x)$ and*

$$\mathrm{BLxy}(h)\backslash\{b(h,1), b(h,2^{h-1}+1)\} \subseteq T \quad \textit{for all } h \geq 2,$$

then T does not belong to $\mathcal{B}(DRec(\Sigma, X))$.

Proof. Suppose that $T \in \mathcal{B}(DRec(\Sigma, X))$. Then there exist $k, n_1, \ldots, n_k \geq 1$ and languages $T_{ij} \in DRec(\Sigma, X)(1 \leq i \leq k \text{ and } 1 \leq j \leq n_i)$, such that

$$\begin{aligned} T \;=\; & (T_{11} \cap T_{12}^C \cap \cdots \cap T_{1n_1}^C) \cup \\ & (T_{21} \cap T_{22}^C \cap \cdots \cap T_{2n_2}^C) \cup \\ & \quad \vdots \\ & (T_{k1} \cap T_{k2}^C \cap \cdots \cap T_{kn_k}^C). \end{aligned}$$

Denote $m = \max_{1 \leq i \leq k} n_i$.

For any $h \geq 2$, the forest $\mathrm{BLxy}(h)\backslash\{b(h,1), b(h,2^{h-1}+1)\}$ is a subset of T and it has $2^{h-1}-1$ elements. Choose then h so big that

$$2^{h-1} - 1 \geq k(m+1).$$

Then there exists an $i \in [1,k]$ such that

$$TB = (\mathrm{BLxy}(h)\backslash\{b(h,1), b(h,2^{h-1}+1)\}) \cap T_{i1} \cap T_{i2}^C \cap \cdots \cap T_{in_i}^C$$

contains at least $m+1$ trees. This means that $|TB| \geq n_i + 1$. Note also that $TB \cap T_{ij} = \emptyset$, if $2 \leq j \leq n_i$.

Consider the set $\Omega(TB)$. Every tree in it has a subtree $\sigma(x,x)$, so no tree in $\Omega(TB)$ belongs to T. Especially, no tree in $\Omega(TB)$ belongs to the set $T_{i1} \cap T_{i2}^C \cap \cdots \cap T_{in_i}^C$.

Let $s = \max\{s_i | b(h,s_i) \in TB\}$. By Lemma 4.1

$$\Omega(TB\backslash\{b(h,s)\}) \subseteq \Delta(TB) \subseteq \Delta(T_{i1}) = T_{i1}.$$

We can not have $n_i = 1$; otherwise $T_{i1} = T_{i1} \cap T_{i_2}^C \cap \cdots \cap T_{in_i}^C$ and the trees in $\Omega(TB\backslash\{b(h,s)\})$ would belong to T. Thus we assume $n_i \geq 2$. Also we can deduce that no tree in $\Omega(TB\backslash\{b(h,s)\})$ belongs to $T_{i2}^C \cap \cdots \cap T_{in_i}^C$.

The injectivity of Ω implies that $|\Omega(TB\backslash\{b(h,s)\})| = |TB\backslash\{b(h,s)\}| = |TB| - 1 \geq n_i$. This means that in $TB\backslash\{b(h,s)\}$ there are two trees $t_1 = b(h,s_1)$ and $t_2 = b(h,s_2)$, where $s_1 < s_2$, and one set T_{ij}^C of the sets $T_{i2}^C, \ldots, T_{in_i}^C$ such that $\Omega(t_1), \Omega(t_2) \notin T_{ij}^C$. In other words, $\Omega(t_1), \Omega(t_2) \in T_{ij}$. By Lemma 4.2

$$t_2 = b(h,s_2) \in \Delta(\{\Omega(t_1), \Omega(t_2)\}) \subseteq \Delta(T_{ij}) = T_{ij}.$$

On the other hand, $t_2 \in TB\backslash\{b(h,s)\}$. Thus $TB \cap T_{ij} \neq \emptyset$, which is a contradiction.

This means that T does not belong to the Boolean closure of DR-recognizable languages. □

Next we study the case where there are no binary operators. Let τ be an m-ary operator for some $m > 2$. First we expand the trees in $\mathrm{BLxy}(h)$ by the following mapping $\Phi : F_{\{\sigma\}}(X) \to F_{\{\tau\}}(X)$:

(1) $\Phi(x) = x$ for all $x \in X$ and
(2) $\Phi(\sigma(t_1, t_2)) = \tau(\Phi(t_1), \Phi(t_2), x, \ldots, x)$.

In fact, Φ is a linear tree homomorphism, but more importantly it is an injection. Moreover, it preserves the height of a tree, and the subtrees of $\Phi(\mathrm{BLxy}(h))$ of height 1 are in the set $\{\tau(x, y, x, \ldots, x), \tau(y, x, x, \ldots, x)\}$. The effect of Φ is illustrated by Figure 1.

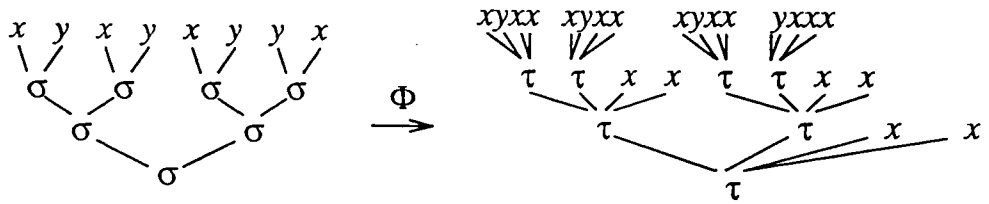

Figure 1. The effect of Φ on the tree $b(3, 4)$.

Lemma 4.4 *Let $m \geq 2, \tau \in \Sigma_m, x, y \in X$ and $T \subseteq F_\Sigma(X)$. If no tree in T has a subtree $\tau(x, x, x, \ldots, x)$ and*

$$\Phi(\mathrm{BLxy}(h)) \backslash \{\Phi(b(h, 1)), \Phi(b(h, 2^{h-1} + 1))\} \subseteq T \quad \textit{for all } h \geq 2,$$

then T does not belong to $\mathcal{B}(DRec(\Sigma, X))$.

Proof. We repeat the proof of Lemma 4.3 using the modified mapping $\Omega : \Phi(\mathrm{BLxy}(h)) \to \Phi(F_\Sigma(X))$, which is defined to replace the leftmost $\tau(y, x, x, \ldots, x)$, by $\tau(x, x, x, \ldots, x)$. If a tree does not have a subtree $\tau(y, x, x, \ldots, x)$, then Ω leaves it unchanged. Also now Ω is an injection in the set $\Phi(\mathrm{BLxy}(h))$.

Throughout Lemma 4.1, Lemma 4.2 and Lemma 4.3 the trees $\Phi(b(h, n))$ are used instead of the trees $b(h, n)$. The proofs of the first two lemmas consider only the ordering of the leaves of subtrees of height 1, and from this point of view the trees $b(h, n)$ and $\Phi(b(h, n))$ are essentially the same.

Lemma 4.3 is based on the fact that $\mathrm{BLxy}(h)$ can always be chosen sufficiently large by increasing h. Because Φ is an injection, the number of trees in $\Phi(\mathrm{BLxy}(h))$ have the same property. Otherwise the rest of the proof continues identically to the proof of Lemma 4.3. □

Theorem 4.5 *If $\Sigma \neq \Sigma_0 \cup \Sigma_1$ and $|X| \geq 2$, then $\mathcal{B}(DRec(\Sigma, X))$ is properly contained in $Rec(\Sigma, X)$. Hence, $\mathcal{B}(DRec)$ is a proper subvariety of Rec.*

Proof. If $\tau \in \Sigma_m$, where $m \geq 2$, and $x, y \in X$, then the ΣX-tree language

$$\begin{aligned} T &= \{t | \{s \in \mathrm{sub}(t) | \mathrm{hg}(s) \leq 1\} \\ &= \{\tau(x, y, x, \ldots, x), \tau(y, x, x, \ldots, x), x, y\}\} \end{aligned}$$

is recognizable, and it satisfies the conditions of Lemma 4.4. Thus it distinguishes the families $\mathcal{B}(DRec(\Sigma, X))$ and $Rec(\Sigma, X)$. □

5 $\mathcal{B}(DRec)$ and Other Varieties

In this section we define the tree language varieties *D*, *RD*, *GD*, *Nil* and *Loc* and compare $\mathcal{B}(DRec)$ with them.

The inclusion relation of varieties is defined componentwise: if $\mathcal{V} = \{\mathcal{V}(X)\}$ and $\mathcal{U} = \{\mathcal{U}(X)\}$ are varieties and $\mathcal{V}(X) \subseteq \mathcal{U}(X)$ for every alphabet X, then we write $\mathcal{V} \subseteq \mathcal{U}$. The trivial variety $Triv = \{\{\emptyset, F_\Sigma(X)\}\}$ and the variety $Rec = \{Rec(\Sigma, X)\}$ of all recognizable languages are the smallest and the largest tree language varieties and $Triv \subset \mathcal{B}(DRec) \subset Rec$. The intersection of varieties $\mathcal{U}$ and $\mathcal{V}$ is $\mathcal{U} \cap \mathcal{V} = \{\mathcal{U}(X) \cap \mathcal{V}(X)\}$.

Definite, reverse definite and generalized definite tree languages were defined by Heuter [Heu89b] and shown to form varieties by Steinby [Ste92].

Definite tree languages. In a definite tree language the membership can be tested by looking at the nodes near the root. These nodes form a part of a tree called the k-root.

The k-root $r_k(t) \in F_\Sigma(X \cup \Sigma) \cup \{\varepsilon\}$ of a tree $t \in F_\Sigma(X)$ is defined as follows:

(1) $r_0(t) = \varepsilon$
(2) $r_1(t) = \mathrm{root}(t)$
(3) Let $k \geq 2$.
  a) If $\mathrm{hg}(t) < k$, then $r_k(t) = t$.
  b) If $\mathrm{hg}(t) \geq k$ and $t = \sigma(t_1, \ldots, t_m)$, then
  $r_k(t) = \sigma(r_{k-1}(t_1), \ldots, r_{k-1}(t_m))$.

The special symbol $\varepsilon \notin X \cup \Sigma$ means the empty tree.

For example, the k-roots of a tree $t = \sigma(\sigma(x, \gamma), y)$ are $r_0(t) = \varepsilon$, $r_1(t) = \sigma$, $r_2(t) = \sigma(\sigma, y)$ and $r_k(t) = t$ for all $k \geq 3$.

Let $k \geq 0$. A forest $T \subseteq F_\Sigma(X)$ is *k-definite*, if for all trees $s, t \in F_\Sigma(X)$,

$$(t \in T \quad \text{and} \quad r_k(s) = r_k(t)) \quad \text{imply} \quad s \in T.$$

The family of all k-definite ΣX-languages is denoted by $D(k, X)$. We write $D(k) = \{D(k, X)\}$. On the other hand, the family of definite Σ-tree languages is $D = \{D(X)\}$, where $D(X) = \bigcup_{k \geq 0} D(k, X)$.

For example, the language $\{\sigma(x, y), \sigma(y, x)\}$ belongs to $D(2, X)$. Note that according to Lemma 2.1 it is not DR-recognizable.

The definition of $D(k, X)$ can be rephrased by means of a congruence θ_k of the term algebra $\mathcal{T}_\Sigma(X)$ which is defined so that, for any ΣX-trees s and t,

$$s \theta_k t \quad \text{iff} \quad r_k(s) = r_k(t).$$

A ΣX-tree language T is k-definite iff it is saturated by θ_k, i.e. $T = T\theta_k$.

The members of a θ_k-class have all the same k-root, which fully determines the class. For a fixed k, there are only finitely many k-roots, and therefore the congruence θ_k is finite.

Reverse definite tree languages. To see whether a tree belongs to a reverse definite tree language only its subtrees lower than given height need to be known.

Let $h \geq 0$ and $t \in F_\Sigma(X)$. Denote by

$$S_h(t) = \{s \in \mathrm{sub}(t) | \mathrm{hg}(s) < h\}$$

all the subtrees of t of height at most $h-1$. A ΣX-tree language is *reverse h-definite*, if for every $s, t \in F_\Sigma(X)$,

$$(t \in T \quad \text{and} \quad S_h(s) = S_h(t)) \quad \text{imply} \quad s \in T.$$

For example, if $t = \sigma(\omega(x), \sigma(x,y))$, then $S_0(t) = \emptyset, S_1(t) = \{x,y\}, S_2(t) = \{\omega(x), \sigma(x,y), x, y\}$ and $S_3(t) = S_4(t) = \cdots = \text{sub}(t)$.

Let $h \geq 0$. The set of all reverse h-definite ΣX-languages is $RD(h,X)$. Also we denote $RD(h) = \{RD(h,X)\}$. The family of all reverse definite Σ-languages is $RD = \{RD(X)\}$, where $RD(X) = \bigcup_{h \geq 0} RD(h,X)$.

As in the case of definite languages there exists a finite congruence θ^h of $\mathcal{F}_\Sigma(X)$ that characterizes the reverse definite ΣX-tree languages. This relation is defined so that, for any $s, t \in F_\Sigma(X)$,

$$s \theta^h t \quad \text{iff} \quad S_h(s) = S_h(t).$$

Now a tree language is reverse h-definite if and only if it is saturated by θ^h.

Generalized definite tree languages. A tree language is generalized definite, if for some $h, k \geq 0$, the membership of a tree is determined only by the tree's k-root and its subtrees of height less than h.

For $h, k \geq 0$ and ΣX-trees s and t, the relation θ_k^h is defined so that

$$s \theta_k^h t \quad \text{iff} \quad (S_h(s) = S_h(t) \quad \text{and} \quad r_k(s) = r_k(t)).$$

Then θ_k^h is a finite congruence of $\mathcal{F}_\Sigma(X)$.

A forest $T \subseteq F_\Sigma(X)$ is *generalized h,k-definite* if and only if for all $s, t \in F_\Sigma(X)$,

$$(t \in T \quad \text{and} \quad s \theta_k^h t) \quad \text{imply} \quad s \in T.$$

Again, a tree language is generalized h,k-definite if and only if it is saturated by the congruence θ_k^h.

The family of all generalized h,k-definite ΣX-tree languages is $GD(h,k,X)$. Then we write $GD(h,k) = \{GD(h,k,X)\}$. Also

$$GD(X) = \bigcup_{h \geq 0} \bigcup_{k \geq 0} GD(h,k,X).$$

Now $GD = \{GD(X)\}$ is the family of all generalized definite ΣX-tree languages.

Comparison between definite varieties and $\mathcal{B}(DRec)$. It is easy to see that $D(0) = RD(0) = GD(0,0) = Triv$. For the general case, the connections between definite, reverse definite and generalized definite tree language families are established by

Theorem 5.1 [Ste92]. *Let $h, k \geq 0$. Then*

(1) $D(k), RD(h), GD(h,k)$ *and* D, RD, GD *are tree language varieties,*
(2) $D(0) \subseteq D(1) \subseteq \cdots \subseteq D \subseteq Rec$,
(3) $RD(0) \subseteq RD(1) \subseteq \cdots \subseteq RD \subseteq Rec$,
(4) $GD(0,k) = D(k)$,
$GD(h,0) = RD(h)$,
(5) $GD(h,k) \subseteq GD(h+1,k) \cap GD(h,k+1)$ *and*
(6) $GD(h,k) \subseteq GD \subseteq Rec$. □

If $\Sigma_0 = \emptyset$ and the ranked alphabet Σ is unary, then every forest is closed under Δ-operation, and $DRec = Rec$. Thus D, RD and GD are all included in $\mathcal{B}(DRec)$. That the inclusion is proper can be seen by considering the forest $T_1 = \{t_i \mid i \text{ is even}\}$, where

(1) $t_0 = x$ and
(2) $t_{n+1} = \sigma(t_n)$.

The language T_1 is DR-recognizable, but it does not belong to any of $D(k, X)$, $RD(h, X)$ or $GD(h, k, X)$ for any $h, k \geq 0$.

If Σ is trivial, then the construction of inclusions of Theorem 5.1 collapses and $Rec = GD(0, 1) = GD(1, 0) = GD(h, k)$ for all $h, k \geq 1$.

We show now that for any Σ with $\Sigma_0 = \emptyset$ the varieties $GD(1, k)$ for every $k \geq 0$, and hence, also varieties D and $RD(1)$ are contained in $\mathcal{B}(DRec)$.

Theorem 5.2 *Let $\Sigma_0 = \emptyset$. For all $k \geq 0$, the variety $GD(1, k)$ is included in $\mathcal{B}(DRec)$.*

Proof. Let X be an alphabet, $k \geq 0$ and $T \in GD(1, k, X)$. Because T is saturated by θ_k^1, it is the union of some θ_k^1-classes. This union is finite, since θ_k^1 is finite. Therefore it suffices to show that any θ_k^1-class belongs to $\mathcal{B}(DRec(\Sigma, X))$.

For $t \in F_\Sigma(X)$, let $t\theta_k^1$ be the θ_k^1-class of t. Because $t\theta_k^1 = t\theta^1 \cap t\theta_k$, we will prove $t\theta_k^1 \in \mathcal{B}(DRec(\Sigma, X))$ by studying $t\theta^1$ and $t\theta_k$ separately.

Firstly, the class $t\theta_k$ is recognizable. If $s \in \Delta(t\theta_k)$, then $r_k(s) = r_k(t)$. So $s \in t\theta_k$. This means $t\theta_k$ is also DR-recognizable.

Secondly, the trees in a θ^1-class have the same set of leaves, which the Δ-operation can only reduce. Thus $t\theta^1$ can be written in the form

$$t\theta^1 = \Delta(t\theta^1) \setminus \bigcup \{\Delta(s\theta^1) \mid \text{leaf}(s) \subset \text{leaf}(t)\}.$$

The sets $\Delta(t\theta^1)$ are DR-recognizable. Namely, if the leaves of t are all the same, say $\text{leaf}(t) = \{x\}$, then $\Delta(t\theta^1) = \{s \mid \text{leaf}(s) = \{x\}\} = t\theta^1$ is recognizable. But if t has at least two different leaves, then

$$\Delta(t\theta^1) = \bigcup \{u\theta^1 \mid \text{leaf}(u) \subseteq \text{leaf}(t)\} \setminus \{s \mid |\delta(s)| = 1\},$$

where both the union of θ^1-classes and the set of *chains* $\{s \mid |\delta(s)| = 1\}$ are recognizable. This means that $t\theta^1 \in \mathcal{B}(DRec(\Sigma, X))$. Hence, $T \in \mathcal{B}(DRec(\Sigma, X))$. □

Next we show by generalizing the previously mentioned forest T_1 that the inclusion of Theorem 5.2 is proper, if Σ is not trivial.

Let $x \in X$ and $\sigma \in \Sigma_m$, for $m \geq 1$. For each $i \geq 0$, we define the special trees s^i so that

(1) $s^0 = \xi$,
(2) $s^1 = \sigma(\xi, x, \ldots, x)$ and
(3) $s^{n+1} = s^n \cdot_\xi \sigma(\xi, x, \ldots, x)$.

Note that the superscript i indicates the height and also the number of σ-nodes in s^i. The forest $T = \{x \cdot_\xi s^i \mid i \text{ is even}\}$ is DR-recognizable, since $\Delta(T) = T$. Lemma 5.3 shows that it is not generalized definite and thus neither definite nor reverse definite.

Lemma 5.3 *The forest* $T = \{x \cdot_\xi s^i \mid i \text{ is even}\}$ *is not generalized* h,k*-definite for any* $h,k \geq 0$.

Proof. Assume that there exist $h,k \geq 1$ such that $T \in GD(h,k,X)$. Now $t = x \cdot_\xi s^{2h+2k} \in T$. Its k-root is $r_k(t) = \sigma \cdot_\xi s^{k-1}$ and $S_h(t) = \{x \cdot_\xi s^i \mid i = 0,1,\ldots,h-1\}$. On the other hand, the tree $u = x \cdot_\xi s^{2h+2k+1}$ has the same k-root as t and the same set of subtrees of height at most $h-1$. Therefore u belongs to T, which is contrary to the definition of T.

If $T \in GD(h,k,X)$ for $h < 1$ or $k < 1$, then $T \in GD(h,k,X)$ for $h,k \geq 1$ by Theorem 5.1 (5). Thus the claim holds for all $h,k \geq 0$. □

As a result we get

Theorem 5.4 *If* $\Sigma \neq \emptyset, \Sigma_0 = \emptyset$ *and* $h,k \geq 0$, *then*

(1) $D \subset \mathcal{B}(DRec)$,
(2) $RD(1) \subset \mathcal{B}(DRec)$,
(3) $GD(1,k) \subset \mathcal{B}(DRec)$,
(4) $\mathcal{B}(DRec) \not\subseteq RD(h)$,
(5) $\mathcal{B}(DRec) \not\subseteq GD(h,k)$,
(6) $\mathcal{B}(DRec) \not\subseteq RD$ *and*
(7) $\mathcal{B}(DRec) \not\subseteq GD$. □

To see that RD and GD are not included in $\mathcal{B}(DRec)$ we recall the language T of Theorem 4.5:

$$T = \{t \in F_\Sigma(X) \mid S_2(t) = \{\tau(x,y,x,\ldots,x), \tau(y,x,x,\ldots,x), x, y\}\}.$$

Now T is a reverse 2-definite tree language, and thus also a generalized definite tree language. Since T does not belong to the Boolean closure of DR-recognizable languages, we have the following

Theorem 5.5 *Let* $\Sigma \neq \Sigma_0 \cup \Sigma_1$. *For* $h \geq 2$ *and* $k \geq 0$, *we have*

(1) $RD(h) \not\subseteq \mathcal{B}(DRec)$,
(2) $GD(h,k) \not\subseteq \mathcal{B}(DRec)$,
(3) $RD \not\subseteq \mathcal{B}(DRec)$ *and*
(4) $GD \not\subseteq \mathcal{B}(DRec)$. □

Finite and cofinite tree languages. The tree language family $Nil = \{Nil(X)\}$ is a variety of Σ-tree languages [Ste92], where the family $Nil(X)$ consists of all finite and cofinite ΣX-tree languages. This variety is contained in both D and RD and it itself contains $Triv$.

Theorem 5.6 *The variety Nil is contained in the variety* $\mathcal{B}(DRec)$. *The inclusion is proper if and only if* $\Sigma \neq \Sigma_0$. □

Local tree languages. Local tree languages are the languages in the Boolean closure of strictly local tree languages. The membership of a tree in a strictly local language is determined, when the root and the forks of a tree are known.

The *forks* of a tree $t \in F_\Sigma(X)$ form a set $\mathrm{fork}(t)$ defined as follows:

(1) If $t \in \Sigma_0 \cup X$, then $\mathrm{fork}(t) = \emptyset$.
(2) If $t = \sigma(t_1,\ldots,t_m)$, then

$$\mathrm{fork}(t) = \{\sigma(\mathrm{root}(t_1), \ldots, \mathrm{root}(t_m))\} \cup \bigcup_{i=1}^{m} \mathrm{fork}(t_i).$$

The set of all forks in ΣX-trees $\mathrm{fork}(\Sigma, X)$ is

$$\mathrm{fork}(\Sigma, X) = \bigcup \{\mathrm{fork}(t) \mid t \in F_\Sigma(X)\}.$$

For example, the forks of $t = \sigma(\omega(x), \sigma(x, y))$ are $\sigma(\omega, \sigma), \omega(x)$ and $\sigma(x, y)$.

A forest $T \subseteq F_\Sigma(X)$ is *local in the strict sense* or *strictly local,* if there exist a set of forks $F \subseteq \mathrm{fork}(\Sigma, X)$ and a set of roots $R \subseteq \Sigma \cup X$ such that

$$t \in T \quad \text{iff} \quad (\mathrm{fork}(t) \subseteq F \quad \text{and} \quad \mathrm{root}(t) \in R).$$

Then we write $T = \mathrm{SL}(R, F)$.

For example, the languages $L_1 = \{\sigma^i \mid i \geq 1\}$ and $L_2 = \{\sigma_i \mid i \geq 1\}$, where σ^i and σ_i denote the trees

$$\sigma^0 = x, \qquad \sigma_0 = x,$$

and

$$\sigma^{n+1} = \sigma(\sigma^n, x) \qquad \sigma_{n+1} = \sigma(x, \sigma_n),$$

are local in the strict sense; they could be defined as $L_1 = \mathrm{SL}(\{\sigma\}, \{\sigma(\sigma, x), \sigma(x, x)\})$ and $L_2 = \mathrm{SL}(\{\sigma\}, \{\sigma(x, \sigma), \sigma(x, x)\})$ as well. But their union $L = L_1 \cup L_2$ is not strictly local. Namely, though the tree $\sigma(x, \sigma(\sigma(x, x), x))$ has σ as root and the forks of it are all forks of trees in L, it does not belong to L.

On the other hand, the intersection $T_1 \cap T_2 = \mathrm{SL}(R_1 \cap R_2, F_1 \cap F_2)$ of two strictly local tree languages $T_1 = \mathrm{SL}(R_1, F_1)$ and $T_2 = \mathrm{SL}(R_2, F_2)$ is strictly local. Note also that $\emptyset = \mathrm{SL}(\emptyset, \emptyset)$ and $F_\Sigma(X) = \mathrm{SL}(\Sigma \cup X, \mathrm{fork}(\Sigma, X))$ are strictly local. However, the previous remarks imply that the complement of a strictly local tree language is not always strictly local.

A forest $T \subseteq F_\Sigma(X)$ is *local,* if it is built from local forests in the strict sense by using finitely many Boolean operations. The family of local Σ-tree languages is $Loc = \{Loc(X)\}$.

The local forests in the strict sense and thereby the local forests are recognizable [GS84]. Furthermore, *Loc* is a tree language variety [Ste92].

Next we show that also the local tree languages have a characterizing family of congruences. It is easy to see that the relation θ defined by

$$s \theta t \quad \text{iff} \quad (\mathrm{root}(s) = \mathrm{root}(t) \quad \text{and} \quad \mathrm{fork}(s) = \mathrm{fork}(t))$$

is a finite congruence.

Lemma 5.7 *Let $L \subseteq F_\Sigma(X)$. Then L is local if and only if θ saturates L.*

Proof. Let $L \in Loc(\Sigma, X)$. Then there exist $k \geq 1$ and $n \geq 2$ such that L can be written in the form

$$\begin{aligned} L \;=\; & (L_{11} \cap L_{12}^C \cap \cdots \cap L_{1n}^C) \cup \\ & (L_{21} \cap L_{22}^C \cap \cdots \cap L_{2n}^C) \cup \\ & \quad \vdots \\ & (L_{k1} \cap L_{k2}^C \cap \cdots \cap L_{kn}^C), \end{aligned}$$

where for every $1 \leq i \leq k$ and $1 \leq j \leq n$, $L_{ij} \in \mathrm{SL}(R_{ij}, F_{ij})$.

Let $t \in L\theta$. Then there exists an $l \in L$ such that $\mathrm{root}(t) = \mathrm{root}(l)$ and $\mathrm{fork}(t) = \mathrm{fork}(l)$. Now

$$l \in L_{i1} \cap L_{i2}^C \cap \cdots \cap L_{in}^C$$

for at least one $i \in [1,k]$. Because $l \in L_{i1}$, then also $t \in L_{i1}$. The reason why $l \notin L_{ij}$ for a $2 \leq j \leq n$ must be either $\mathrm{root}(l) \notin R_{ij}$ or $\mathrm{fork}(l) \not\subseteq F_{ij}$. In both cases also $t \notin L_{ij}$. Together this means $t \in L$. So $L = L\theta$.

Conversely assume $L = L\theta$. This means that L is the union of some θ-classes. To show that L is local one only needs to verify that any θ-class is local. It is, because if $l \in L$, then

$$l\theta = \mathrm{SL}(\mathrm{root}(l), \mathrm{fork}(l)) \setminus \bigcup \{\mathrm{SL}(\mathrm{root}(l), F) | F \subset \mathrm{fork}(l)\}.$$

□

Now we are ready to compare *Loc* with *DRec* and $\mathcal{B}(DRec)$.

Theorem 5.8 *If $\Sigma \neq \Sigma_0$, we have*

(1) $DRec \not\subseteq Loc$,
(2) $\mathcal{B}(DRec) \not\subseteq Loc$, *and*
(3) $Loc \subset Rec$.

Proof. Let $x \in X \cup \Sigma_0$ and $\sigma \in \Sigma_m$ for $m \geq 1$. The DR-recognizable forest $T = \{\sigma(\sigma(x, \ldots, x), x, \ldots, x)\}$ is not local, since θ does not saturate it. □

The tree language $T_1 = \{\sigma(x,y), \sigma(y,x)\}$ is not DR-recognizable by Lemma 2.1, but clearly it is local. Hence, $Loc \not\subseteq DRec$. However, T_1 does belong to the Boolean closure of DR-recognizable languages. So the question now is, whether this holds for all local languages. For this purpose we consider the following language.

Let $\sigma \in \Sigma_m$, where $m \geq 2$, and $x, y \in X$. Define F as the set of forks $F = \{\sigma(\sigma, \sigma, x, \ldots, x), \sigma(x, y, x, \ldots, x), \sigma(y, x, x, \ldots, x)\}$. Then the forest $\mathrm{SL}(\{\sigma\}, F)$ satisfies the conditions of Lemma 4.4 and thus does not belong to the Boolean closure of DR-recognizable languages. This leads us to

Theorem 5.9 *Let $\Sigma \neq \Sigma_0 \cup \Sigma_1$. Then*

(1) $Loc \not\subseteq DRec$ *and*
(2) $Loc \not\subseteq \mathcal{B}(DRec)$. □

If Σ is unary and $\Sigma_0 = \emptyset$, then *Loc* is contained in $DRec = Rec$. Theorem 5.8 shows that this inclusion is proper. But if Σ is trivial, then every language is local and $Loc = DRec = Rec$.

Figure 2 shows the inclusion relations of varieties for $\Sigma_0 = \emptyset$. If also $\Sigma \neq \Sigma_0 \cup \Sigma_1$, the inclusions are proper and those varieties not connected are incomparable.

Acknowledgment I am grateful to Professor Magnus Steinby for his expert guidance and valuable suggestions during the course of this work.

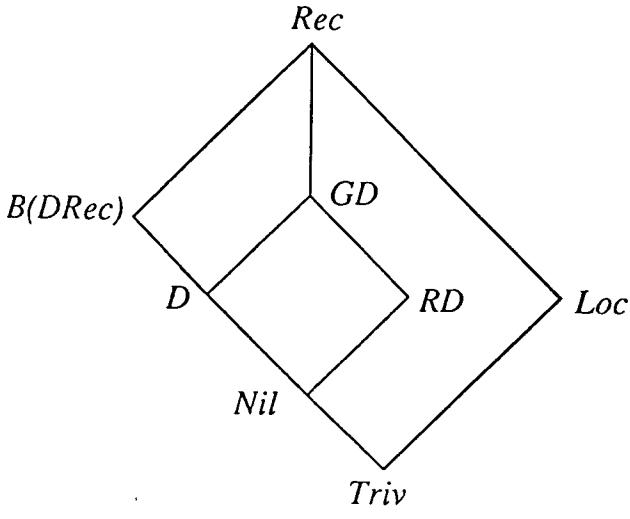

Figure 2. Comparation of studied varieties.

References

[Cou78a] Courcelle, B., A representation of trees by languages I, Theoret. Comput. Sci. 6 (1978), 255-279.

[Cou78b] Courcelle, B., A representation of trees by languages II, Theoret. Comput. Sci. 7 (1978), 25-55.

[Don70] Doner, J.E., Tree acceptors and some of their applications, J. Comput. Syst. Sci. 4 (1970), 406-451.

[GS78] Gécseg, F. and Steinby M., Minimal ascending tree automata, Acta Cybernet. 4 (1978), 37-44.

[GS84] Gécseg, F. and Steinby M., Tree automata, Akadémiai Kiadó, Budapest, 1984.

[Heu88] Heuter, U., Definite tree languages, Bull. EATCS 35 (1988), 137-144.

[Heu89a] Heuter, U., Generalized definite tree languages, Mathem. Found. Comput. Sci. (Proc. Symp., Porabka-Kozubnik, Poland 1989) Lect. Notes in Comput. Sci. 379, Springer-Verlag, Berlin, 1989, pp. 270-280.

[Heu89b] Heuter, U., Zur Klassifizierung regulärer Baumsprachen, Dissertation, Technical University of Aachen, Aachen, 1989.

[MM69] Magidor, M. and Moran, G., Finite automata over finite trees, Technical Report 30, Hebrew University, Jerusalem, 1969.

[Rab69] Rabin, M.O., Decidability of second-order theories and automata on infinite trees, Trans. Am. Math. Soc. 141 (1969), 1-35.

[Sik64] Sikorski, R., Boolean algebras, Springer-Verlag, Berlin, 1964.

[Ste79] Steinby, M., Syntactic algebras and varieties of recognizable sets, Les arbres en algèbre et en programmation, 4ème Coll. Lille (Proc. Coll., Lille 1979), Université de Lille, 1979, pp. 226-240.

[Ste92] Steinby, M., A theory of tree language varieties, Tree Automata and Languages (Nivat, M. and Podelski, A., eds.), Elsevier Publishers, Amsterdam, 1992, pp. 57-81.

[Tha73] Thatcher, J. W., Tree automata: an informal survey, Currents in the Theory of Computing (Aho, A.V., ed.), Prentice-Hall, Englewood Cliffs, N.J., 1973, pp. 143-172.

[TW68] Thatcher, J. W. and Wright, J. B., Generalized finite automata theory with an application to a decision problem of second order logic, Math. Syst. Theory 2 (1968), 57-81.

[Tho84] Thomas, W., Logical aspects in the study of tree languages, Ninth colloquium on trees in algebra and programming (Courcelle, B., ed.), Cambridge Univ. Press, Cambridge, 1984, pp. 31-51.

[Vir80] Virágh, J., Deterministic ascending tree automata I, Acta Cybernet. 5 (1980), 33-42.

Received August 4, 1992.

Acta Cybernetica, Vol. 10, Nr. 4, Szeged, 1992

A Queuing Model for a Processor-Shared Multi-Terminal System Subject to Breakdowns

B. Almási*

Abstract

This paper deals with a non-homogeneous finite-source queuing model to describe the performance of a multi-terminal system subject to random breakdowns under PPS (Priority Processor-Sharing) service discipline. It can be viewed as a continuation of paper [1], which discussed a FIFO (First-In, First-Out) serviced queuing model subject to random breakdowns. All random variables are assumed to be independent and exponentially distributed. The system's behaviour can be described by a Markov chain, but the number of states is very large (it is a combinatorically increasing function of the number of terminals). The purpose of this paper is to give a recursive computational approach to solve the steady-state equations and to illustrate the problem in question using some numerical results.

1 The Model

This paper deals with a terminal system consisting of n terminals connected with a Central Processor Unit (CPU). The user at the terminal i thinks for random times and generates jobs to the CPU. The think times are assumed to be exponentially distributed with mean $\frac{1}{\lambda_i}$. The required running times of jobs of terminal i are exponentially distributed random variables with mean $\frac{1}{\mu_i}$ (assuming, that the jobs use the whole capacity of the CPU). The jobs staying at the CPU are serviced in parallel using the PPS scheduling strategy (see [2,3]). Each terminal has a positive weight, denoted by ω_i for terminal $i (i = 1, \ldots, n)$, and if there are $s (1 \leq s \leq n)$ jobs at the CPU from the terminals $j_1, \ldots, j_s$ then the job of the terminal $j_r (r = 1, \ldots, s)$ is serviced at rate

$$W_{j_r}(j_1, \ldots, j_s) = \frac{\omega_{j_r}}{\sum_{i=1}^{s} \omega_{j_i}},$$

that is, the processing intensity is $W_{j_r}(j_1, \ldots, j_s)\mu_{j_r}$ for the job of terminal j_r. Let us suppose that the CPU is subject to random breakdowns stopping the whole system. The failure-free operation times of the CPU are exponentially distributed random variables with mean $\frac{1}{\alpha}$. The restoration times of the CPU are exponentially

*Department of Mathematics, University of Debrecen, Debrecen P.O.Box 12, H-4010, Hungary
Acknowledgement: The author is very grateful to Professor M. Arató and J. Sztrik for their helpful comments. Research is partially supported by Hungarian National Foundation for Scientific Research under grant OTKA 1648/91.

distributed with mean $\frac{1}{\beta}$. The busy terminals are also subject to random breakdowns not affecting the system's operation but stopping the work at the terminal. The failure-free operation times of busy terminals are supposed to be exponentially distributed random variables with mean $\frac{1}{\gamma_i}$ for terminal i. The repair times of terminal i are exponentially distributed random variables with mean $\frac{1}{\tau_i}$. The breakdowns are serviced by a single repairman according to FIFO discipline among terminals and providing preemptive priority to the failure of CPU. We assume that each terminal sleeps while its job is serviced by the CPU, that is, the terminal is inactive while waiting at the CPU, and it cannot break down. All random variables involved here are assumed to be independent of each other.

On the one hand this paper is a generalization of the non-homogeneous PS model discussed in [4] (which allowed only CPU failures), on the other hand it further generalizes the homogeneous model treated in [5] (which allowed both terminal and CPU failures). This paper is the continuation of [1] where the FIFO discipline was discussed (instead of PPS) and we build a new non-homogeneous model and solve the steady-state equations recursively by using a similar computational approach as in [1]. In equilibrium the main performance of the system, such as the mean number of jobs residing at the CPU, the mean number of functional terminals, the expected response time of jobs, and utilizations are obtained. Finally it is investigated - by using some numerical results - how breakdowns affect the performance characteristics and the results of [1] are compared with ours.

2 The Mathematical Model and a Computational Approach

Let us introduce the following random variables:

$$X(t) = \begin{cases} 1, & \text{if the CPU is failed at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

$Y(t) =$ the failed terminals' indices at time t in order of their failure, or 0 if there is no failed terminal,

$Z(t) =$ the indices of jobs staying at the CPU at time t in lexicographically increasing order, or 0 if there is no job at the CPU.

It is easy to see that the process

$$M(t) = (X(t), Y(t), Z(t)),$$

is a multi-dimensional Markov chain with 3 vector-components and with state space

$$S = ((q; i_1, \ldots, i_k; j_1, \ldots, j_s),\ q = 0, 1;\ k = 0, \ldots, n;\ s = 0, \ldots, n-k),$$

where

$(i_1, \ldots, i_k)$ is a permutation of K objects from the numbers $1, \ldots, n$ or 0, if $k = 0$,

$(j_1, \ldots, j_s)$ is a combination of s objects from the remaining $n-k$ numbers or 0, if $s = 0$.

The event $(q; i_1, \ldots, i_k; j_1, \ldots, j_s)$ denotes that the operating system is in state $X(t) = q$, there are k failed terminals with indices $i_1, \ldots, i_k$, and there are s jobs with indices $j_1, \ldots, j_s$ at the CPU $(i_r \neq j_l,\ r = 1, \ldots, k;\ l = 1, \ldots, s)$.

The reader can easily verify that the number of states is

$$\dim(S) = 2 \sum_{k=0}^{n} \sum_{s=0}^{k} \frac{n!}{(n-k)!s!}.$$

Let us denote the steady-state distribution of $(M(t), t \geq 0)$ by

$$p(q; i_1, \ldots, i_k; j_1, \ldots, j_s) = \lim_{t \to \infty} p(X(t) = q; Y(t) = i_1, \ldots, i_k; Z(t) = j_1, \ldots, j_s),$$

which exists and is unique (see [6]) because of all the rates are assumed to be positive. For brevity let us introduce the following notation:

$$K_{j_r}(q; i_1, \ldots, i_k; j_1, \ldots, j_s) = \frac{\omega_{j_r} \mu_{j_r}}{\sum_{i=1}^{s} \omega_{j_i}} p(q; i_1, \ldots, i_k; j_1, \ldots, j_s),\ r = 1, \ldots, s.$$

Since we study the steady-state behavior of the Markov chain $M(t)$, following [6], we can start with the statement

$$\text{Average rate of leaving state } (q; i_1, \ldots, i_k; j_1, \ldots, j_s) =$$

$$= \text{Average rate of entering state } (q; i_1, \ldots, i_k; j_1, \ldots, j_s),$$

that is, we can build the global balance equations for $p(q i_1, \ldots, i_k; j_1, \ldots, j_s)$ by using the rules discussed in Section 1:

$$(\alpha + \tau_{i_1} + \sum_{\substack{r \neq i_1, \ldots, i_k \\ r \neq j_1, \ldots, j_s}} (\lambda_r + \gamma_r)) p(0; i_1, \ldots, i_k; j_1, \ldots, j_s) + \sum_{r=1}^{s} K_{j_r}(0; i_1, \ldots, i_k; j_1, \ldots, j_s) =$$

$$= \beta p(1; i_1, \ldots, i_k; j_1, \ldots, j_s) + \tag{1}$$

$$+ \sum_{\substack{r \neq i_1, \ldots, i_k \\ r \neq j_1, \ldots, j_s}} (\tau_r p(0; r, i_1, \ldots, i_k; j_1, \ldots, j_s) + K_r(0; i_1, \ldots, i_k; j_1, \ldots, r, \ldots, j_s)) +$$

$$+ \gamma_{i_k} p(0; i_1, \ldots, i_{k-1}; j_1, \ldots, j_s) + \sum_{r=1}^{s} \lambda_{j_r} p(0; i_1, \ldots, i_k; j_1, \ldots, j_{r-1}, j_{r+1}, \ldots, j_s),$$

$$\text{for all } i_1, \ldots, i_k; j_1, \ldots, j_s; k = 0, \ldots, n; s = 0, \ldots, n-k,$$

$$\beta p(1; i_1, \ldots, i_k; j_1, \ldots, j_s) = \alpha p(0; i_1, \ldots, i_k; j_1, \ldots, j_s), \tag{2}$$

$$\text{for all } i_1, \ldots, i_k; j_1, \ldots, j_s; k = 0, \ldots, n; s = 0, \ldots, n-k,$$

where the probabilities of meaningless events and coefficients are defined to be zero.

For $k = 0$ and $s = 0$ i_k (and j_s) are not defined, so for example the element $\gamma_{i_k} p(0; i_1, \ldots, i_{k-1}; j_1, \ldots, j_s)$ has no meaning, so it is defined to be zero, we have:

$$(\alpha + \sum_{i=1}^{n}(\lambda_i + \gamma_i))p(0;0;0) = \beta p(1;0;0) + \sum_{i=1}^{n} \mu_i p(0;0;i) + \sum_{i=1}^{n} \tau_i p(0;i;0).$$

The system of equations will be simpler if we substitute Equation (2) to Equation (1). Namely, we have

$$\begin{aligned}
(\tau_{i_1} &+ \sum_{\substack{r \neq i_1, \ldots, i_k \\ r \neq j_1, \ldots, j_s}} (\lambda_r + \gamma_r))p(0; i_1, \ldots, i_k; j_1, \ldots, j_s) + \\
&+ \sum_{r=1}^{s} K_{j_r}(0; i_1, \ldots, i_k; j_1, \ldots, j_s) = \\
&= \sum_{\substack{r \neq i_1, \ldots, i_k \\ r \neq j_1, \ldots, j_s}} (\tau_r p(0; r, i_1, \ldots, i_k; j_1, \ldots, j_s) + K_r(0; i_1, \ldots, i_k; j_1, \ldots, r, \ldots, j_s)) + \\
&+ \gamma_{i_k} p(0; i_1, \ldots, i_{k-1}; j_1, \ldots, j_s) \\
&+ \sum_{r=1}^{s} \lambda_{j_r} p(0; i_1, \ldots, i_k; j_1, \ldots, j_{r-1}, j_{r+1}, \ldots, j_s),
\end{aligned}$$

$$\text{for all } i_1, \ldots, i_k; j_1, \ldots, j_s; k = 0, \ldots, n; s = 0, \ldots, n-k, \tag{3}$$

$$\beta p(1; i_1, \ldots, i_k; j_1, \ldots, j_s) = \alpha p(0; i_1, \ldots, i_k; j_1, \ldots, j_s), \tag{4}$$

$$\text{for all } i_1, \ldots, i_k; j_1, \ldots, j_s; k = 0, \ldots, n; s = 0, \ldots, n-k.$$

The purpose of this paper is to solve this system subject to the normalization condition

$$\sum_{q=0}^{1} \sum_{k=0}^{n} \sum_{s=0}^{n-k} p(q, k, s) = 1,$$

where

$$p(q, k, s) = \sum_{(i_1, \ldots, i_k) \in V_n^k} \sum_{(j_1, \ldots, j_s) \in C_{n-k}^s} p(q; i_1, \ldots, i_k; j_1, \ldots, j_s),$$

V_n^k : The set of all $(i_1, \ldots, i_k)$ (as defined above),
C_{n-k}^s : the set of all $(j_1, \ldots, j_s)$ (as defined above).

Such a system of linear equations could easily be solved by standard computational methods, e.g., by Gauss- elimination. But we must not forget that the unknowns are probabilities and therefore - since the state space is very large - the round off errors may have considerable effect on them (see [7,8,9]) and when using computer programs to solve the system of equations, the whole matrix of the equations cannot be stored in a personal computer if $n \geq 3$. It is more efficient to use a recursive computational method to determine the steady-state probabilities, as described in the following section (first it was proposed by Tomkó [10]).

3 The Recursive Solution

Let $\underline{Y}(m)$ be the vector of the stationary probabilities for the states where the operating system is working, there are k failed terminals, and $s = m - k$ jobs are waiting at the CPU $((k = 0, \ldots, m), m = 0, \ldots, n)$. That is,

$$\underline{Y}(m) = \begin{pmatrix} p(0;1,\ldots,m-1,m;0) \\ p(0;1,\ldots,m-1,m+1;0) \\ \vdots \\ p(0;n,\ldots,n-m+1;0) \\ p(0;1,\ldots,m-1;m) \\ p(0;1,\ldots,m-1;m+1) \\ \vdots \\ p(0;n,\ldots,n-m+2;n-m+1) \\ \vdots \\ \vdots \\ p(0;0;n-m+1,\ldots,n) \end{pmatrix}$$

In words, the elements of $\underline{Y}(m)$ are written in lexicographically increasing order of indices

1./ for $k = m$ and $s = 0$,
2./ for $k = m - 1$ and $s = 1$,
$\vdots$
$m + 1$./ for $k = 0$ and $s = m$.

Similarly, let $\underline{Z}(m)$ be the vector of stationary probabilities alike $\underline{Y}(m)$, but for the states, where the CPU is failed. From the definition it is obivous that the dimension of $\underline{Y}(m)$ and $\underline{Z}(m)$ is $\sum_{s=0}^{m} \frac{n!}{(n-m)!s!}$.

Using these notations Equations (3), (4) can be written in matrix form as

$$\underline{Y}(0) = C(0)\underline{Y}(1), \qquad (i)$$
$$\underline{Y}(j) = C(j)\underline{Y}(j+1) + D(j)\underline{Y}(j-1),\ j = 1,\ldots,n-1, \qquad (ii)$$
$$\underline{Y}(n) = D(n)\underline{Y}(n-1), \qquad (iii)$$
$$\underline{Z}(j) = F(j)\underline{Y}(j),\ j = 0,\ldots,n. \qquad (iv)$$

The dimension of the matrices are / $d(j) = \sum_{s=0}^{j} \frac{n!}{(n-j)!s!}$ / :
$F(j) : d(j) \times d(j)$, $C(j) : d(j) \times d(j+1)$, $D(j) : d(j) \times d(j-1)$.
The elements of all the matrices can be obtained from the Equations (3), (4). For example we can use Equation (4) to obtain the elements of matrix $F(k+s)(k+s = 0,\ldots,n)$: The element $p(1;i_1,\ldots i_k;j_1,\ldots,j_s)$ of $Z(k+s)$ can be obtained from the element $p(0;i_1,\ldots,i_k;j_1,\ldots,j_s)$ of $Y(k+s)$ by multiplying it with $\frac{\alpha}{\beta}$. That is, the matrix $F(k+s)$ contains non-zero elements only in its main diagonal, and this non zero element is the constant value $\frac{\alpha}{\beta}$.

Similarly, we can use the second line of Equation (3) to build the matrix $C(k+s)$, and the third line to determine the matrix $D(k+s)(k+s = 0,\ldots,n)$.
Applying these notations we can state our main result:

Theorem 3.1 *The solution of the Equations (i)-(iv) can be given in the form*

$$\underline{Y}(j) = L(j)\underline{Y}(j-1), j = 1, \ldots, n,$$

$$\underline{Z}(j) = F(j)\underline{Y}(j), j = 0, \ldots, n, \tag{5}$$

where $L(n) = D(n)$, $L(j) = (I - C(j)L(j+1))^{-1}D(j)$, $j = 1, \ldots, n-1$, *so the system of equations can be solved uniquely up to a multiplicative constant, which can be obtained from the normalization condition.*

Proof. As a starting point of our proof we can observe that equation (iv) is identical with the second equation of (5).
In virtue of equation (iii)

$$\underline{Y}(n) = L(n)Y(n-1).$$

Assuming that $\underline{Y}(j+1) = L(j+1)\underline{Y}(j)$, from (ii) we have

$$\underline{Y}(j) = C(j)L(j+1)\underline{Y}(j) + D(j)\underline{Y}(j-1).$$

By simple calculation we obtain that

$$(I - C(j)L(j+1))\underline{Y}(j) = D(j)\underline{Y}(j-1),$$

$$\underline{Y}(j) = (I - C(j)L(j+1))^{-1}D(j)\underline{Y}(j-1),$$

$$\underline{Y}(j) = L(j)\underline{Y}(j-1),$$

which completes the proof.

Now we can start the recursion with any initial value denoted by $\underline{Y'}(0)$ and the non-normalized $p'(q; i_1, \ldots, i_k; j_1, \ldots, j_s)$ elements of $\underline{Y'}(m)$, $\underline{Z'}(m)(m = 0, \ldots, n)$, can be obtained. We can calculate the steady-state probabilities from $\underline{Y'}(m)$, $\underline{Z'}(m)$ $(m = 0, \ldots, n)$, by using the normalization condition as follows

$$\underline{Y}(m) = \frac{\underline{Y'}(0)}{\sum_{q=0}^{1}\sum_{k=0}^{n}\sum_{s=0}^{n-k}\sum_{i_1,\ldots,i_k \in V_n^k}\sum_{j_1,\ldots,j_s \in C_{n-k}^s} p'(q; i_1, \ldots, i_k; j_1, \ldots, j_s)}\underline{Y'}(m),$$

$$\underline{Z}(m) = \frac{\underline{Y'}(0)}{\sum_{q=0}^{1}\sum_{k=0}^{n}\sum_{s=0}^{n-k}\sum_{i_1,\ldots,i_k \in V_n^k}\sum_{j_1,\ldots,j_s \in C_{n-k}^s} p'(q; i_1, \ldots, i_k; j_1, \ldots, j_s)}\underline{Z'}(m),$$

$m = 0, \ldots, n.$

4 Performance Measures

We derive the steady-state characteristics from the steady-state probabilities because the model is too complicated to derive the characteristics directly from the parameters $(n, \alpha, \beta, \ldots)$. Some of these characteristics will be calculated in Table 1-5 (for $n = 4$ and $n = 3$) as examples. We can use these numerical results to investigate how parameters influence the characteristics.

We employ the following usual notation: $\delta(i, j) = 1$, if $i = j$, (and 0 otherwise). The steady-state characteristics:

(i) Mean number of jobs residing at the CPU

$$\overline{n}_j = \sum_{i=0}^{1} \sum_{k=0}^{n} \sum_{s=0}^{n-k} sp(i, k, s).$$

(ii) Mean number of functional terminals

$$\overline{n}_g = n - \sum_{i=0}^{1} \sum_{k=0}^{n} \sum_{s=0}^{n-k} kp(i, k, s).$$

(iii) Mean number of busy terminals

$$\overline{n}_b = \sum_{k=0}^{n} \sum_{s=0}^{n-k} (n - k - s)p(0, k, s).$$

(iv) Utilization of repairman

$$U_r = \sum_{k=0}^{n} \sum_{s=0}^{n-k} p(1, k, s) + \sum_{k=1}^{n} \sum_{s=0}^{n-k} p(0, k, s).$$

(v) Utilization of CPU

$$U_{CPU} = \sum_{k=0}^{n-1} \sum_{s=1}^{n-k} p(0, k, s).$$

(vi) Utilization of terminal $i/i = 1, \ldots, n/$

$$U_i = \sum_{k=0}^{n} \sum_{s=0}^{n-k} \sum_{r=1}^{k} \sum_{v=1}^{s} (1 - \delta(i, i_r) - \delta(i, j_v))p(0; i_1, \ldots, i_k; j_1, \ldots, j_s).$$

(vii) Expected response time of jobs for terminal $i/i = 1, \ldots, n/$

$$T_i = \frac{\sum_{q=0}^{1} \sum_{s=0}^{n-k} \sum_{r=1}^{s} \delta(i, j_r)p(q; i_1, \ldots, i_k; j_1, \ldots j_s)}{\lambda_i U_i}.$$

5 Numerical Results

The algorithm described above was implemented on an IBM PC/AT in FORTRAN. We show several examples to illustrate how breakdowns influence the characteristics. The running times were at about 50 seconds for $n = 4$ (Table 1-3), and 2 seconds for $n = 3$ (Table 4-5). If we compare these results to the ones described in [1,10] we can see how scheduling strategy influences the characteristics.

Case 1. Failure free system (See [10]).

$n = 4$ $\alpha = 0.0001$ $\beta = 9999.0$
$\overline{n}_j = 2.195$ $\overline{n}_g = 4.0$ $U_{CPU} = 0.906$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.500	0.900	0.0001	9999.0	1.0	0.429	2.658
2	0.400	0.700	0.0001	9999.0	1.0	0.423	3.405
3	0.300	0.600	0.0001	9999.0	1.0	0.452	4.045
4	0.200	0.500	0.0001	9999.0	1.0	0.500	4.998

Table 1.

This case will be the starting point of our investigation. It can be used to test the results and to approximate a failure-free system described in [10]. The difference between these results and the ones in [10] is less than 0.01 for all calculated characteristics. The difference can be derived from the different calculating circumstances (e.g. different computer and programming language). On the other hand this case only approximates a failure-free system.

Case 2. Terminal failure.

$n = 4$ $\alpha = 0.0001$ $\beta = 9999.0$
$\overline{n}_j = 1.253$ $\overline{n}_g = 2.58$ $U_{CPU} = 0.666$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.500	0.900	0.3200	0.4500	1.0	0.283	2.133
2	0.400	0.700	0.1700	0.3400	1.0	0.335	2.602
3	0.300	0.600	0.2200	0.5000	1.0	0.336	3.138
4	0.200	0.500	0.1600	0.3000	1.0	0.373	3.842

Table 2.

In this example we can see how terminal failures influence the performance measures. The response times and the number of good terminals are less than in Case 1. That is, the system works as if there were less terminals.

Case 3. CPU failure.

$n = 4 \quad \alpha = 0.25 \quad \beta = 0.45$
$\overline{n}_j = 2.195 \quad \overline{n}_g = 4.0 \quad U_{CPU} = 0.583$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.500	0.900	0.0001	9999.0	1.0	0.276	4.135
2	0.400	0.700	0.0001	9999.0	1.0	0.272	5.296
3	0.300	0.600	0.0001	9999.0	1.0	0.290	6.292
4	0.200	0.500	0.0001	9999.0	1.0	0.321	7.775

Table 3.

If we compare these results with Case 1, it can be seen, that the failure of the CPU increases the response times and decreases the utilizations, as one can expect.

It seems, that the FIFO (see in [1]) and the PS discipline gives nearly the same results investigating the influence of terminal breakdowns. We got greater CPU utilization in this model than in [1] (for each case). This is the reason why this model is more sensible to the CPU breakdowns (see the response times in Case 3).

Case 4. PPS system (see [3]).

$n = 3 \quad \alpha = 0.0001 \quad \beta = 9999.0$
$\overline{n}_j = 1.028 \quad \overline{n}_g = 3.0 \quad U_{CPU} = 0.675$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.200	0.400	0.0001	9999.0	1.0	0.508	4.831
2	0.200	0.600	0.0001	9999.0	5.0	0.666	2.498
3	0.200	0.800	0.0001	9999.0	125.0	0.796	1.277

Table 4.

This case can be used to test the algorithm (and the computer program) discussed above. A failure-free PPS system (described in [3]) is approximated by this example. The results are exactly the same as in [3].

Case 5. PPS system with CPU failure.

$n = 3 \quad \alpha = 0.1000 \quad \beta = 0.7000$
$\overline{n}_j = 1.028 \quad \overline{n}_g = 3.0 \quad U_{CPU} = 0.591$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.200	0.400	0.0001	9999.0	1.0	0.445	5.521
2	0.200	0.600	0.0001	9999.0	5.0	0.583	2.855
3	0.200	0.800	0.0001	9999.0	125.0	0.696	1.459

Table 5.

This case shows, that the more a terminals uses the CPU (or the CPU's queue) the more the response time increases with the CPU failure.

Case 6. PPS system for $n = 5$.

$n = 5$ $\alpha = 0.1000$ $\beta = 0.7000$
$\overline{n}_j = 2.278$ $\overline{n}_g = 4.3$ $U_{CPU} = 0.777$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.200	0.400	0.1000	0.7000	1.0	0.217	15.6472
2	0.300	0.700	0.0700	0.6000	5.0	0.319	5.836
3	0.250	0.650	0.1500	0.4500	50.0	0.418	2.951
4	0.220	0.600	0.1000	0.7600	15.0	0.406	4.706
5	0.400	0.800	0.1200	0.4400	125.0	0.442	1.747

Table 6.

The program was run for $n = 5$ in this case. This was the largest n value that could be used. The running time was at about 4 minutes.

References

[1] B. Almási and J. Sztrik, A Queueing Model for a Non-Homogeneous Terminal System Subject to Breakdowns, *Computers and Mathematics with Applications* (to appear).

[2] E. Gelenbe and I. Mitrani, *Analysis and Synthesis of Computer Systems,* Academic Press, London, (1980).

[3] J. Sztrik, A probability model for priority processor-shared multiprogrammed computer systems, *Acta Cybernetica 7,* 329-340 (1986).

[4] J. Sztrik, On the heterogeneous machine interference with limited server's availability, *European Journal of Op. Res. 28,* 321-328 (1987).

[5] J. Sztrik and T. Gál, A recursive solution of a queueing model for a multi-terminal system subject to breakdowns, *Performance Evaluation 11,* 1-7 (1990).

[6] R. Goodman, *Introduction to Stochastic Models,* Benjamin/Cummings, California, (1988).

[7] S.S. Lavenberg, Ed., *Computer Performance Modelling Handbook,* Academic Press, New York, (1983).

[8] P.M. Snyder and W.J. Stewart, Explicit and iterative approaches to solving queueing models, *Oper. Res. 33,* 183-202 (1985).

[9] H.C. Tijms, *Stochastical Modelling and Analysis, A Computational Approach,* Wiley and Sons, New York, (1986).

[10] L. Csige and J. Tomkó, The machine interference for exponentially distributed operating and repair times, *Alk. Mat. Lapok 8,* 107-124 (in Hungarian) (1982).

Received September 29, 1992.

Acta Cybernetica, Vol. 10, Nr. 4, Szeged, 1992

The Self-organizing List and Processor Problems under Randomized Policies

T. Makjamroen*

Abstract

We consider the self-organizing list problem in the case that only one item has a different request probability and show that transposition has a steady state cost stochastically smaller than any randomized policy that moves the requested item, found in position i, to position j with some probability $a_{ij}, i \geq j$. A random variable X is said to be stochastically smaller than another random variable Y, written $X \leq_{st} Y$ if $Pr\{X \geq k\} \leq Pr\{Y \geq k\}$, for any k. This is a stronger statement than $\mathrm{E}[X] \leq \mathrm{E}[Y]$. We also show that the steady state cost under the policy that moves the requested item i positions forward is stochastically increasing in i. Sufficient conditions are given for the steady state cost under a randomized policy **A** to be stochastically smaller than that under another randomized policy **B**. Similar results are obtained for the processor problem, where a list of processors is considered.

OPTIMAL LIST ORDER; MEMORY CONSTRAINTS; TRANSPOSITION RULE; RAMDOMIZATION

0 Introduction

A *self-organizing list problem* is characterized by a sequential list of n items subject to a reordering policy. At the beginning of each time period, an item is requested and the list is searched sequentially from the first position until the requested item is found. Each of these n items has an unknown probability of being requested. Let $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ be the request probability vector, where p_i is the request probability of item $i, i = 1, \ldots, n$, and $0 < p_i \leq 1$, $\sum_{i=1}^{n} p_i = 1$. At the end of each period, the items on the list are reordered according to the reordering policy. The cost of each period is taken to be the position where the requested item is found. We are interested in the steady state costs under various policies. A reordering policy is called *optimal* if it minimizes the expected steady state cost for any given request probability vector $\mathbf{p}$. The self-organizing list problem will now be called *the list problem* and the *policy* will mean the reordering policy.

Kan and Ross [6] define a *no-memory* policy as a reordering policy that depends only on the position of the requested item and the current ordering. Some of the most studied examples of the no-memory policies are the *transposition, move-to-front,* and *move-i-position* policies. Keeping the relative positions of all other

*Department of Economics, Thammasat University, Bangkok, Thailand 10200

items unchanged, the move-i-position policy moves the requested item i positions closer to the front if the requested item is found at position $j, j > i$, otherwise the requested item is moved to the first position. Transposition is just move-1-position and move-to-front is move-$(n-1)$-position for a problem of n items. Hendricks [3,4] gives the steady state probability distributions of states under move-to-front and transposition. See Hester and Hirschberg [5] for a recent survey of the list problem.

Anderson, Nash and Weber [1] show by counterexample that transposition is not optimal. However, their counterexample not only moves the requested item but also changes the positions of other items. So it is still an open question if transposition is optimal among policies that move only the requested item, leaving the relative ordering of the rest unchanged.

In the special case where only one item has a different request probability, Kan and Ross [6] and Phelps and Thomas [7] show that transposition is indeed optimal among policies that move only the requested item. We will show in Section 1.2 that transposition is optimal in a stronger sense. In particular, by extending the induction argument used by Phelps and Thomas, we can show that transposition has a steady state cost *stochastically smaller* than that of any *randomized policy*. Let $C(\mathbf{p};\mathbf{A})$ be the steady state cost of the list problem with request probability vector $\mathbf{p}$ under policy $\mathbf{A}$. Then $C(\mathbf{p};\mathbf{A})$ is stochastically smaller than $C(\mathbf{p};\mathbf{B})$, written $C(\mathbf{p};\mathbf{A}) \leq_{st} C(\mathbf{p};\mathbf{B})$, if $Pr\{C(\mathbf{p};\mathbf{A}) \geq k\} \leq Pr\{C(\mathbf{p};\mathbf{B}) \geq k\}, k = 1, 2, \ldots, n$. It follows immediately that $\mathrm{E}[C(\mathbf{p};\mathbf{A})] \leq \mathrm{E}[C(\mathbf{p};\mathbf{B})]$. A randomized policy is a policy which, when an item is requested and found at position i, moves that item to position j with some probability $a_{ij}, \sum_{j=1}^{i} a_{ij} = 1$, leaving the relative ordering of others unchanged.

Section 1.1 defines the randomized policy and shows its properties. By the introduction of the randomized policy, we also show in Section 1.2 that move-i-position has a steady state cost stochastically increasing in i. This partially supports the conjecture of Gonnett, Munro, and Suwanda [2]. Their conjecture says that if $\mathbf{A}$ and $\mathbf{B}$ are two no-memory policies such that if the requested item is found at position i, it is moved forward $A(i)$ and $B(i)$ positions by the policies $\mathbf{A}$ and $\mathbf{B}$ respectively, and $A(i) \leq B(i), i = 1, \ldots, n$, then the expected steady state cost under $\mathbf{A}$ is smaller than or equal to that under $\mathbf{B}$, but $\mathbf{B}$ converges to its asymptotic behavior more quickly than $\mathbf{A}$. Furthermore, it also follows that if the cost is taken to be an increasing function of the position where the requested item is found, move-i-position will have an expected steady state cost increasing in i. A special case of this situation is found in the *paging problem* as also discussed by Phelps and Thomas [7] where for a fixed integer $m, 1 \leq m \leq n$, the cost is taken to be zero if the requested item is found in a postion less than m, and one otherwise.

Tenenbaum and Nemes [9] consider two spectra of policies. Assuming that only one item has a different request probability, the policies in each of the two spectra are ordered by the values of their expected steady state costs. Each spectrum has transposition at one end with the minimum expected steady state cost and move-to-front at the other with the maximum expected steady state cost. We will show in Section 1.2 that the steady state costs of these policies in each spectrum are stochastically smaller or larger than each other.

A problem related to the list problem is called *the processor problem* which was studied by, among others, Topkis [10]. In the processor problem, we consider a sequential list containing an ordering of the n processors. Each of these processors has an unknown probability that it will successfully process a given job. At the beginning of each time priod, there is an arrival of a job to be processed. The job is attempted by the processors successively according to the ordering in the list until either one of the processors succeeds or all of them fail. Then the job is dismissed. The cost in each period is taken to be the number of processors attempted until

the job is processed, or, in the case that all n processors fail, it is taken to be n. At the end of each period, a reordering policy is applied in the same manner as in the list probelm. For example, we might move the successful processor to the beginning of the ordering (move-to-front), or we might just move it one position closer to the front (transposition).

Topkis [10] gives the steady state probabilities of the move-to-front and move-to-back policies and shows that move-to-front has a steady state cost stochastically smaller than move-to-back, which in turn, has a steady state cost stochastically smaller than the random policy where processors are equally likely to be in any of the $n!$ orderings.

Section 2.1 shows the properties of randomized policy when applied to the processor problem with only one processor having a different success probability. In this special case, Ross [8] shows that the expected steady state cost under transposition is smaller than or equal to that under move-to-front. In Section 2.2, we also use randomized policies to obtain results closely parallel to those of the list problem. That is, the steady state cost under transposition is stochastically smaller than that under any randomized policy. Furthermore, the steady state cost under move-i-position is stochastically smaller than that under move-$(i+1)$-position. The steady state costs under the policies in the two spectra proposed by Tenenbaum and Nemes [9] are also ordered such that the steady state cost of each policy is stochastically smaller or larger than its neighbors in the same spectrum.

1 The List Problem

When only item 1 has a different request probability, the expected steady state cost can be written in terms of the expected position of the item 1. That is, by conditioning on whether item 1 is being requested,

$$\begin{aligned} \mathrm{E}[C(\mathbf{p};\mathbf{A})] &= cp\mathrm{E}[Y_1(\mathbf{p};\mathbf{A})] + \frac{p(n-1)\mathrm{E}[1+2+\cdots+n-Y_1(\mathbf{p};\mathbf{A})]}{n-1} \\ &= p(c-1)\mathrm{E}[Y_1(\mathbf{p};\mathbf{A})] + \frac{pn(n+1)}{2}, \end{aligned}$$

where $Y_1(\mathbf{p};\mathbf{A})$ is the steady state position of item 1 of the list problem with request probability vector $\mathbf{p}$ under policy $\mathbf{A}$, $p_1 = cp, p_2 = p, \ldots, p_n = p$, and $c > 0$.

So when $c > 1$, we want to minimize $\mathrm{E}[Y_1(\mathbf{p};\mathbf{A})]$, and maximize it when $c < 1$. For the rest of the paper, we assume that $c > 1$. The results for $c < 1$ will be just the opposite.

1.1 Randomized Policy

A randomized policy is characterized by a matrix $\mathbf{A} = [A_{ij}]_{n\times n}$, where $A_{ij} = \sum_{k=1}^{j} a_{ik}$, and a_{ij} is the probability that given an item is requested and found at position i, it is moved to position j, where $\sum_{j=1}^{i} a_{ij} = 1$ for all i, and $0 \le a_{ij} \le 1$. So A_{ij} is the probability that given the requested item is found at position i, it is moved to a position less than or equal to j.

Given a policy $\mathbf{A}$ defined in a system of n items, define a related policy $\mathbf{A}^d$ in a system of $n-1$ items as follows.

$$\mathbf{A}^d = [A^d_{ij}]_{(n-1)\times(n-1)},$$

where $A_{ij}^d = \sum_{k=1}^{j} a_{ik}^d$, and

$$a_{ij}^d = \begin{cases} a_{i+1,1} + a_{i+1,2} & , j = 1 \\ a_{i+1,j+1} & , j \geq 2. \end{cases} \tag{1.1}$$

Let π_i^A be the steady state probability that item 1 is at position i under policy **A**. That is, $\pi_i^A = \Pr\{Y_1(\mathbf{p};\mathbf{A}) = i\}$. Alternatively, we can say $Y_1(\mathbf{p};\mathbf{A}) \leq_{st} Y_1(\mathbf{p};\mathbf{B})$ by using the notation $\{\pi_i^A\} \leq_{st} \{\pi_i^B\}$. Define $K_i^A = \pi_i^A/\pi_n^A$. Lemma 1.1 to Lemma 1.4 below show the relationships between $\{\pi_i^A\}$ and $\{\pi_i^{dA}\}$ under the assumption that $(p_1^d, p_2^d, \ldots p_{n-1}^d) = (cp^d, p^d, \ldots, p^d)$. Lemma 1.1 and Lemma 1.2 are also obtained by Phelps and Thomas [7], where they consider only policies that move the requested item, found at position i, to a fixed position $\tau(i), \tau(i) \leq i$.

Lemma 1.1 *Under policy* **A**, *for* $i = 2, \ldots, n$,

$$\pi_i^A/\pi_n^A = K_i^A = \pi_{i-1}^{dA}/\pi_{n-1}^{dA}.$$

Proof. The transition matrix, showing only columns $1, r+1$ and n can be written as

$$\begin{bmatrix} cp + p\sum_{i=2}^{n}\sum_{j=2}^{i} a_{ij} & \cdots & 0 & \cdots & 0 \\ cpa_{21} & \cdots & \vdots & \cdots & \vdots \\ cpa_{31} & \cdots & 0 & \cdots & \\ \vdots & \cdots & p\sum_{i=r+1}^{n}\sum_{j=1}^{r} a_{ij} & \cdots & \\ & \cdots & cpa_{r+1,r+1} + p\theta & \cdots & \vdots \\ & \cdots & cpa_{r+2,r+1} & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & p(1-a_{nn}) \\ cpa_{n1} & \cdots & cpa_{n,r+1} & \cdots & cpa_{nn} + p(n-1) \end{bmatrix} \tag{1.2}$$

where $\theta = r + \sum_{i=r+2}^{n}\sum_{j=r+2}^{i} a_{ij}$

Except the first column, column $r+1$ contains zeros from row 1 to row r. Using the $(r+1)^{st}$ column of the transition matrix and suppressing the superscript **A**, we have

$$\begin{aligned} \pi_{r+1} &= \pi_r p\left(\sum_{i=r+1}^{n}\sum_{j=1}^{r} a_{ij}\right) + \pi_{r+1}\left[cpa_{r+1,r+1} + p\theta\right] \\ &\quad + \sum_{i=r+2}^{n} cpa_{i,r+1}\pi_i, \end{aligned}$$

for $r = 1, \ldots, n-1$.

Since $p = \frac{1}{c+n-1}$, the above equation becomes

$$\begin{aligned}
&\pi_r\left(\sum_{i=r+1}^{n}\sum_{j=1}^{r} a_{ij}\right)\\
&= \pi_{r+1}\left[c+n-1-ca_{r+1,r+1}-\left(r+\sum_{i=r+2}^{n}\sum_{j=r+2}^{i} a_{ij}\right)\right]-\sum_{i=r+2}^{n} ca_{i,r+1}\pi_i\\
&= \pi_{r+1}\left[n+c(1-a_{r+1,r+1})-(r+1)-\sum_{i=r+2}^{n}\sum_{j=r+2}^{i} a_{ij}\right]-c\sum_{i=r+2}^{n} a_{i,r+1}\pi_i,
\end{aligned} \tag{1.3}$$

where $r = 1,\ldots,n-1$.

For policy $\mathbf{A}^d$, using the r^{th} column of the transition matrix of $n-1$ items and noting that $p^d = \frac{1}{c+n-2}$, we have in the same manner as (1.3) above

$$\begin{aligned}
&\pi^d_{r-1}\left(\sum_{i=r}^{n-1}\sum_{j=1}^{r-1} a^d_{ij}\right)\\
&= \pi^d_r\left[c+n-2-ca^d_{rr}-\left(r-1+\sum_{i=r+1}^{n-1}\sum_{j=r+1}^{i} a^d_{ij}\right)\right]-\sum_{i=r+1}^{n-1} ca^d_{ir}\pi^d_i\\
&= \pi^d_r\left[n+c(1-a^d_{rr})-(r+1)-\sum_{i=r+1}^{n-1}\sum_{j=r+1}^{i} a^d_{ij}\right]-c\sum_{i=r+1}^{n-1} a^d_{ir}\pi^d_i,
\end{aligned} \tag{1.4}$$

where $r = 2,\ldots,n-1$. By the definition of a^d_{ij} given in (1.1), (1.4) becomes

$$\begin{aligned}
&\pi^d_{r-1}\left(\sum_{i=r+1}^{n}\sum_{j=1}^{r} a_{ij}\right)\\
&= \pi^d_r\left[n+c(1-a_{r+1,r+1})-(r+1)-\sum_{i=r+2}^{n}\sum_{j=r+2}^{i} a_{ij}\right]-c\sum_{i=r+2}^{n} a_{i,r+1}\pi^d_{i-1},
\end{aligned} \tag{1.5}$$

where $r = 2,\ldots,n-1$. From (1.3) and (1.5), $(\pi_2,\ldots,\pi_n)$ and $(\pi^d_1,\ldots,\pi^d_{n-1})$ satisfy the same set of equations. We will use this fact to show that $K_i = K^d_{i-1}, i = 2,\ldots,n$, and this proves the Lemma. Since $K_n = K^d_{n-1} = 1$ by definition, we use the induction hypothesis that $K_i = K^d_{i-1}, i = r+1,\ldots,n$. We will show that it is also true for $i = r$. But this follows immediately by dividing both sides of (1.3) and (1.5) by π_n and π^d_{n-1} respectively. □

If we know π^A_i, $i = 1,\ldots,n$, then we know π^{dA}_i, $i = 1,\ldots,n-1$. The exact relationship is given in Lemma 1.2.

Lemma 1.2 *Under policy* **A**, *for* $i = 2, 3, \ldots, n$,

$$\pi_i^A = (1 - \pi_1^A)\pi_{i-1}^{dA}.$$

Proof. From Lemma 1.1, we need to show that $\pi_n^A = (1-\pi_1^A)\pi_{n-1}^{dA}$. By suppressing superscript **A**,

$$1 = \sum_{i=1}^{n-1} \pi_i^d = \sum_{i=1}^{n-1} K_i^d \pi_{n-1}^d = \pi_{n-1}^d \sum_{i=2}^{n} K_i = \pi_{n-1}^d (1 - \pi_1)/\pi_n.$$

□

Conversely, given π_i^{dA}, $i = 1, \ldots, n-1$, we can compute π_i^A, $i = 1, \ldots, n$, using Lemma 1.2 and the following Lemma 1.3.

Lemma 1.3 *Under policy* **A**,

$$\pi_1^A = \frac{c(a_{21}\pi_2^A + a_{31}\pi_3^A + \cdots + a_{n1}\pi_n^A)}{a_{21} + a_{31} + \cdots + a_{n1}}.$$

Proof. The Lemma is proved by using the first column of the transition matrix (1.2) and noting that $p = \frac{1}{c+n-1}$. □

From Lemma 1.1 and Lemma 1.3, Lemma 1.4 below says that we can write K_j in terms of $K_{j+1}, K_{j+2}, \ldots, K_n$. Note that $A_{i1} = a_{i1}, i = 2, \ldots, n$. So Lemma 1.3 and Lemma 1.4 are equivalent when $j = 1$.

Lemma 1.4 *Under policy* **A**, *for* $j = 1, 2, \ldots, n-1$,

$$K_j^A = \frac{c(A_{j+1,j}K_{j+1}^A + A_{j+2,j}K_{j+2}^A + \cdots + A_{nj}K_n^A)}{A_{j+1,j} + A_{j+2,j} + \cdots + A_{nj}}.$$

Proof. From Lemma 1.1, $K_2 = K_1^d, K_3 = K_2^d = K_1^{d^2}, \ldots, K_j = K_1^{d^{j-1}}$. By exactly the same argument, we have $K_k^{d^{j-1}} = K_{k+1}^{d^{j-2}} = \cdots = K_{j+k-1}, k = 2, \ldots, n-j+1$. From Lemma 1.3,

$$K_1^{d^{j-1}} = \frac{c(a_{21}^{d^{j-1}} K_2^{d^{j-1}} + a_{31}^{d^{j-1}} K_3^{d^{j-1}} + \cdots + a_{n-j+1,1}^{d^{j-1}} K_{n-j+1}^{d^{j-1}})}{a_{21}^{d^{j-1}} + a_{31}^{d^{j-1}} + \cdots + a_{n-j+1,1}^{d^{j-1}}}.$$

Now, by definiton (1.1),

$$\begin{aligned} a_{21}^{d^{j-1}} &= a_{31}^{d^{j-2}} + a_{32}^{d^{j-2}} \\ &= a_{41}^{d^{j-3}} + a_{42}^{d^{j-3}} + a_{43}^{d^{j-3}} \\ &\vdots \\ &= a_{j+1,1} + a_{j+1,2} + \ldots + a_{j+1,j} \\ &= A_{j+1,j}. \end{aligned}$$

Similarly, $a_{k1}^{d^{j-1}} = A_{j+k-1,j}, k = 3, \ldots, n-j+1$. So follows the Lemma. □

1.2 Comparison of the Steady State Costs and Probabilities of Two Lists under Two Different Policies

Let S be the set of policies that the resulting probability distribution $\{\pi_i\}$ is decreasing in i when $p_1 > p$ and increasing in i otherwise. The question of how to determine if a policy is in S will be addressed later. We are now ready to prove the following Theorem that compares $\{\pi_i\}$ of two different policies.

Theorem 1.5 *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two policies such that, for* $j = 1, 2, \ldots, n-1, k = j+1, \ldots, n$,

$$\frac{A_{j+1,j} + A_{j+2,j} + \cdots + A_{kj}}{A_{j+1,j} + A_{j+2,j} + \cdots + A_{nj}} \geq \frac{B_{j+1,j} + B_{j+2,j} + \cdots + B_{kj}}{B_{j+1,j} + B_{j+2,j} + \cdots + B_{nj}}, \tag{1.6}$$

and at least one of these two conditions holds:

(a) $\mathbf{A} \in S$ *and* B_{ij} *is decreasing in* i *for all* $j = 1, \ldots, n$,
(b) $\mathbf{B} \in S$ *and* A_{ij} *is decreasing in* i *for all* $j = 1, \ldots, n$.

Then $\{\pi_i^A\} \leq_{st} \{\pi_i^B\}$ *for any* $\mathbf{p} = (cp, p, \ldots, p), c > 1$.

Proof. We will prove this Theorem by induction. It is easily checked that the Theorem is true for $n = 2$. Assume that it is true for the problem of $n-1$ items. Now given such policies $\mathbf{A}$ and $\mathbf{B}$, their corresponding policies $\mathbf{A}^d$ and $\mathbf{B}^d$ also satisfy all the conditions above. We can check this by first noting by that by (1.1)

$$A^d_{ij} = a^d_{i1} + a^d_{i2} + \cdots + a^d_{ij} = a_{i+1,1} + a_{i+1,2} + a_{i+1,3} + \cdots + a_{i+1,j+1} = A_{i+1,j+1}.$$

Therefore, A^d_{ij} is also decreasing in i, and

$$\begin{aligned}\frac{A_{j+1,j} + A_{j+2,j} + \cdots + A_{kj}}{A_{j+1,j} + A_{j+2,j} + \cdots + A_{nj}} &= \frac{A^d_{j,j-1} + A^d_{j+1,j-1} + \cdots + A^d_{k-1,j-1}}{A^d_{j,j-1} + A^d_{j+1,j-1} + \cdots + A^d_{n-1,j-1}} \\ &\geq \frac{B^d_{j,j-1} + B^d_{j+1,j-1} + \cdots + B^d_{k-1,j-1}}{B^d_{j,j-1} + B^d_{j+1,j-1} + \cdots + B^d_{n-1,j-1}}.\end{aligned}$$

Secondly, since $\mathbf{A} \in S, \pi_1^A \geq \pi_2^A \geq \cdots \geq \pi_n^A$. But from Lemma 1.2, $\pi_i^{dA} = \frac{\pi_{i+1}^A}{1-\pi_1^A}$, so $\pi_1^{dA} \geq \pi_2^{dA} \geq \cdots \geq \pi_{n-1}^{dA}$. This means $\mathbf{A}^d \in S$. So we have the induction hypothesis that

$$\{\pi_i^{dA}\} \leq_{st} \{\pi_i^{dB}\}.$$

From Lemma 1.2, $\pi_i^A + \pi_{i+1}^A + \cdots + \pi_n^A = (1-\pi_1^A)(\pi_{i-1}^{dA} + \pi_i^{dA} + \cdots + \pi_{n-1}^{dA})$. All we need to show is that $\pi_1^A \geq \pi_1^B$. From Lemma 1.2 and Lemma 1.3,

$$\pi_1^A = c(1-\pi_1^A)\frac{A_{21}\pi_1^{dA} + A_{31}\pi_2^{dA} + \cdots + A_{n1}\pi_{n-1}^{dA}}{A_{21} + A_{31} + \cdots + A_{n1}}.$$

Since $\pi_1^A \geq \pi_1^B$ if and only if $\frac{\pi_1^A}{1-\pi_1^A} \geq \frac{\pi_1^B}{1-\pi_1^B}$, we need to show that

$$\frac{A_{21}\pi_1^{dA} + A_{31}\pi_2^{dA} + \cdots + A_{n1}\pi_{n-1}^{dA}}{A_{21} + A_{31} + \cdots + A_{n1}} \geq \frac{B_{21}\pi_1^{dB} + B_{31}\pi_2^{dB} + \cdots + B_{n1}\pi_{n-1}^{dB}}{B_{21} + B_{31} + \cdots + B_{n1}}.$$

Assume first that (a) holds. Then, by (1.6) with $j = 1$ and, because $\mathbf{A}^d \in S$, $\pi_1^{dA} \geq \pi_2^{dA} \geq \cdots \geq \pi_{n-1}^{dA}$,

$$\begin{aligned}\frac{A_{21}\pi_1^{dA} + A_{31}\pi_2^{dA} + \cdots + A_{n1}\pi_{n-1}^{dA}}{A_{21} + A_{31} + \cdots + A_{n1}} &\geq \frac{B_{21}\pi_1^{dA} + B_{31}\pi_2^{dA} + \cdots + B_{n1}\pi_{n-1}^{dA}}{B_{21} + B_{31} + \cdots + B_{n1}} \\ &\geq \frac{B_{21}\pi_1^{dB} + B_{31}\pi_2^{dB} + \cdots + B_{n1}\pi_{n-1}^{dB}}{B_{21} + B_{31} + \cdots + B_{n1}}.\end{aligned}$$

The second inequality follows from the assumption that B_{i1} is decreasing in i and from the induction hypothesis that $\{\pi_i^{dA}\} \leq_{st} \{\pi_i^{dB}\}$.

Similarly, if (b) holds,

$$\begin{aligned}\frac{A_{21}\pi_1^{dA} + A_{31}\pi_2^{dA} + \cdots + A_{n1}\pi_{n-1}^{dA}}{A_{21} + A_{31} + \cdots + A_{n1}} &\geq \frac{A_{21}\pi_1^{dB} + A_{31}\pi_2^{dB} + \cdots + A_{n1}\pi_{n-1}^{dB}}{A_{21} + A_{31} + \cdots + A_{n1}} \\ &\geq \frac{B_{21}\pi_1^{dB} + B_{31}\pi_2^{dB} + \cdots + B_{n1}\pi_{n-1}^{dB}}{B_{21} + B_{31} + \cdots + B_{n1}}\end{aligned}$$

□

A consequence of this Theorem is that the steady state cost under policy $\mathbf{A}$ is stochastically smaller than the steady state cost under policy $\mathbf{B}$.

Corollary 1.6 *Under the conditions of Theorem 1.5,* $C(\mathbf{p};\mathbf{A}) \leq_{st} C(\mathbf{p};\mathbf{B})$.

Proof. Conditioning on whether item 1 is at the first position, for $k = 2, \ldots, n$,

$$\begin{aligned}&\Pr\Big\{C(\mathbf{p};\mathbf{A} \geq k\Big\} \\ &= \pi_1^A \Pr\Big\{C(\mathbf{p};\mathbf{A}) \geq k | Y_1(\mathbf{p};\mathbf{A}) = 1\Big\} + (1-\pi_1^A)\Pr\Big\{C(\mathbf{p};\mathbf{A}) \geq k | Y_1(\mathbf{p};\mathbf{A}) \neq 1\Big\} \\ &= \pi_1^A \Pr\Big\{C(\mathbf{p};\mathbf{B}) \geq k | Y_1(\mathbf{p};\mathbf{B}) = 1\Big\} + (1-\pi_1^A)\Pr\Big\{C(\mathbf{p};\mathbf{A}) \geq k | Y_1(\mathbf{p};\mathbf{A}) \neq 1\Big\}.\end{aligned}$$

Now given that item 1 is not at position 1, the probability that it will be at position i, $2 \leq i \leq n$, is $\frac{\pi_i^A}{1-\pi_1^A}$, which is exactly π_{i-1}^{dA} by Lemma 1.2. That is, given item 1 is not at position 1, its probability distribution over $\{2, 3, \ldots, n\}$ is the same as the probability distribution of $\{\pi_i^{dA}\}$ over $\{1, 2, \ldots n-1\}$. Using the induction hypothesis that the Corollary is true for the list of size $n-1$, we have

$$\begin{aligned}\Pr\Big\{C(\mathbf{p};\mathbf{A}) \geq k | Y_1(\mathbf{p};\mathbf{A}) \neq 1\Big\} &= (1-p)\Pr\Big\{C(\mathbf{p}^d;\mathbf{A}^d) \geq k-1\Big\} \\ &\leq (1-p)\Pr\Big\{C(\mathbf{p}^d;\mathbf{B}^d) \geq k-1\Big\} \\ &= \Pr\Big\{C(\mathbf{p};\mathbf{B}) \geq k | Y_1(\mathbf{p};\mathbf{B}) \neq 1\Big\}.\end{aligned}$$

Therefore,

$$\begin{aligned}
& \Pr\Big\{C(\mathbf{p};\mathbf{A}) \geq k\Big\} \\
\leq\ & \pi_1^A \Pr\Big\{C(\mathbf{p};\mathbf{B}) \geq k | Y_1(\mathbf{p};\mathbf{B}) = 1\Big\} + (1-\pi_1^A) \Pr\Big\{C(\mathbf{p};\mathbf{B}) \geq k | Y_1(\mathbf{p};\mathbf{B}) \neq 1\Big\} \\
\leq\ & \pi_1^B \Pr\Big\{C(\mathbf{p};\mathbf{B}) \geq k | Y_1(\mathbf{p};\mathbf{B}) = 1\Big\} + (1-\pi_1^B) \Pr\Big\{C(\mathbf{p};\mathbf{B}) \geq k | Y_1(\mathbf{p};\mathbf{B}) \neq 1\Big\} \\
=\ & \Pr\Big\{(\mathbf{p};\mathbf{B}) \geq k\Big\}.
\end{aligned}$$

The second inequality follows from the fact that $\pi_1^A \geq \pi_1^B$ and, when $p_1 > p$, $\Pr\Big\{C(\mathbf{p};\mathbf{B}) \geq k | Y_1(\mathbf{p};\mathbf{B}) = 1\Big\} \leq \Pr\Big\{C(\mathbf{p};\mathbf{B}) \geq k | Y_1(\mathbf{p};\mathbf{B}) \neq 1\Big\}$. □

By Lemma 1.2 and Corollary 1.6, transposition is optimal in the sense that it has a steady state cost stochastically smaller than any randomized policy. Let T denote the transposition policy.

Corollary 1.7 *For any policy* $\mathbf{A}, C(\mathbf{p};T) \leq_{st} C(\mathbf{p};\mathbf{A})$.

Proof. Given $c > 1$, Phelps and Thomas [7] show that $\pi_1^T \geq \pi_1^Z$ for any policy $\mathbf{Z}$ that moves the requested item strictly forward by using the fact that $\pi_i^Z = (1-\pi_1^Z)\pi_{i-1}^{dZ}$. Since this fact also holds for any randomized policy $\mathbf{A}$ as shown in Lemma 1.2, so $\pi_1^T \geq \pi_1^A$ and thus $\{\pi_i^T\} \leq_{st} \{\pi_i^A\}$ by the same induction argument in Theorem 1.5. The Corollary then follows by Corollary 1.6. □

The next question is how we know if $\mathbf{A} \in S$. The counterexample below shows that not every policy $\mathbf{A}$ is in S even with A_{ij} nonincreasing in i for all j.

A counterexample:

Let A be a policy characterized by the following matrix.

$$\mathbf{A} = \begin{bmatrix}
1 & 0 & 0 & & & 0 & 0 & 0 \\
\varepsilon & 1 & 0 & & & \vdots & \vdots & \vdots \\
\varepsilon & 1-\varepsilon & 1 & \ddots & & & & \\
\vdots & \varepsilon & 1-\varepsilon & \ddots & & & & \\
 & \vdots & \varepsilon & \ddots & \ddots & & & \\
 & & \vdots & \ddots & \ddots & 0 & & \\
 & & & & \ddots & 1 & 0 & \vdots \\
\vdots & \vdots & \vdots & & \ddots & 1-\varepsilon & 1 & 0 \\
\varepsilon & \varepsilon & \varepsilon & & & \varepsilon & 1-\varepsilon & 1
\end{bmatrix}$$

Let ε be some small number. The policy $\mathbf{A}$ almost always moves the requested item one position closer unless the requested item is founded at position 2 where it stays put with probability $1-\varepsilon$ and moves to position 1 with probability ε. By selecting small enough ε, we can get the values of K_i, as given by Lemma 1.4, to approach c^{n-i} arbitrarily close for $i \geq 2$. The value of K_1, as also given by Lemma

1.4, is

$$\begin{aligned} K_1 &= \frac{c(\varepsilon c^{n-2} + \varepsilon c^{n-3} + \cdots + \varepsilon c + \varepsilon)}{\varepsilon + \varepsilon + \cdots + \varepsilon} \\ &= \frac{c(c^{n-1} - 1)}{(n-1)(c-1)}. \end{aligned}$$

With $c = 3$ and $n = 6$, $K_1 = 72.6$ while $K_2 = 3^4 = 81$. So here K_i is not decreasing in i when $c > 1$. Thus not every policy has $\{\pi_i\}$ decreasing in i when $c > 1$. ■

The following Proposition gives a sufficient condition for $\mathbf{A} \in S$. This sufficient condition turns out to be true for any policy $\mathbf{A}$ under which the distribution of the number of positions to move the requested item is independent of the position where it is found. In other words, there is only one distribution for all positions. Call these policies *position independent*. One can interpret a position independent policy as one that uses a mixture of move-i-position, $i = 1, \ldots, n-1$.

Proposition 1.8 *A policy* $\mathbf{A} \in S$ *if, for* $j = 1, \ldots, n-1$,

$$\frac{A_{j+1,j} + \cdots + A_{nj}}{A_{j+2,j+1} + \cdots + A_{n,j+1}} \leq \frac{A_{j+1,j} + \cdots + A_{n-1,j}}{A_{j+2,j+1} + \cdots + A_{n-1,j+1}} \leq \cdots \leq \frac{A_{j+1,j} + A_{j+2,j}}{A_{j+2,j+1}}. \tag{1.7}$$

Proof. Since $A^d_{ij} = A_{i+1,j+1}$, a condition similar to (1.7) holds for $\mathbf{A}^d$. By the induction hypothesis, $\mathbf{A}^d \in S$ and $\pi_1^{dA} \geq \pi_2^{dA} \geq \cdots \geq \pi_{n-1}^{dA}$. So by using Lemma 1.1 we have $\pi_2^A \geq \pi_3^A \geq \cdots \geq \pi_n^A$ and $K_2^A \geq K_3^A \geq \cdots \geq K_n^A$. Thus it remains to show that $\pi_1^A \geq \pi_2^A$. By Lemma 1.3, this means we have to show

$$\frac{A_{21}K_2^A + A_{31}K_3^A + \cdots + A_{n1}K_n^A}{A_{32}K_3^A + A_{42}K_4^A + \cdots + A_{n2}K_n^A} \geq \frac{A_{21} + A_{31} + \cdots + A_{n1}}{A_{32} + A_{42} + \cdots + A_{n2}}.$$

Rewrite the nominator on the left hand side of the above inequality as follows.

$$\begin{aligned} A_{21}K_2^A + A_{31}K_3^A + \cdots + A_{n1}K_n^A &= K_n^A(A_{21} + A_{31} + \cdots + A_{n1}) \\ &+ (K_{n-1}^A - K_n^A)(A_{21} + A_{31} + \cdots + A_{n-1,1}) + \cdots \\ &\cdots + (K_3^A - K_4^A)(A_{21} + A_{31}) + (K_2^A - K_3^A)A_{21} \end{aligned}$$

The left hand side of the last inequality becomes

$$\frac{K_n^A(A_{21} + A_{31} + \cdots + A_{n1}) + (K_{n-1}^A - K_n^A)(A_{21} + A_{31} + \cdots + A_{n-1,1}) + \cdots}{K_n^A(A_{32} + A_{42} + \cdots + A_{n2}) + \cdots}$$

$$\frac{\cdots + (K_3^A - K_4^A)(A_{21} + A_{31}) + (K_2^A - K_3^A)A_{21}}{\cdots + (K_{n-1}^A - K_n^A)(A_{32} + A_{42} + \cdots + A_{n-1,2}) + \cdots + (K_3^A - K_4^A)A_{32}},$$

and because $K_i^A - K_{i+1}^A \geq 0$, $i = 2, \ldots, n-1$, it is greater than the right hand side if

$$\frac{A_{21} + A_{31} + \cdots + A_{n1}}{A_{32} + A_{42} + \cdots + A_{n2}} \leq \frac{A_{21} + A_{31} + \cdots + A_{n-1,1}}{A_{32} + A_{42} + \cdots + A_{n-1,2}} \leq \cdots \leq \frac{A_{21} + A_{31}}{A_{32}},$$

which is just (1.7) with $j = 1$. This follows from the fact that, $\frac{a}{b} \leq \frac{a+c}{b+d}$ if $\frac{a}{b} \leq \frac{c}{d}$, where a, b, c and d are positive. □

We will show next that (1.7) holds for any position independent policy that moves, with probability $a_i, \sum_{i=0}^{n-1} a_i = 1$, the requested item i positions forward if it is found at a position greater than or equal to $i+1$. Otherwise the policy moves the requested item to the first position. Thus, $a_{ij} = a_{i-j}, j > 1$, and $a_{i1} = a_{i-1} + a_i + \cdots + a_{n-1}$. Let $\overline{A}_i = \sum_{k=i}^{n-1} a_k$ be the probability that the requested item is moved more than or equal to i positions. Thus,

$$A_{ij} = a_{i1} + a_{i2} + \cdots + a_{ij} = (a_{i-1} + \cdots + a_{n-1}) + a_{i-2} + \cdots + a_{i-j} = \overline{A}_{i-j}.$$

So (1.7) becomes

$$\frac{\overline{A}_1 + \overline{A}_2 + \cdots + \overline{A}_{n-1}}{\overline{A}_1 + \overline{A}_2 + \cdots + \overline{A}_{n-2}} \leq \frac{\overline{A}_1 + \overline{A}_2 + \cdots + \overline{A}_{n-2}}{\overline{A}_1 + \overline{A}_2 + \cdots + \overline{A}_{n-3}} \leq \cdots \leq \frac{\overline{A}_1 + \overline{A}_2}{\overline{A}_1},$$

which can be shown to be true by just cross-multiplying terms on each side of each inequality and noting that $\overline{A}_i$ is decreasing in i by its definition. Thus we have proved the following Lemma.

Lemma 1.9 *Let* **A** *be a position independent policy that moves requested item i positions with probability a_i, $\sum_{i=0}^{n-1} a_i = 1$. Then* $\mathbf{A} \in S$.

When $a_0 > 0$, we can look at the embedded Markov chain when the items actually change positions. The probability that item 1 is at position i in this embedded Markov chain will be equal to the proportion of time item 1 is at position i in the original chain. The policy governing the embedded chain is characterized by $a_i^* = \frac{a_i}{1-a_0}, i = 1, \ldots, n$, and $a_0^* = 0$. We can, without loss of generality, restrict ourselves from now on to the position independent policies that always move the requested item at least one position closer to the front, unless it is already at the first position.

When two position independent policies **A** and **B** are compared, (1.7) of Proposition 1.8 becomes, for $k = 1, \ldots, n-1$,

$$\frac{\overline{A}_1 + \overline{A}_2 + \cdots + \overline{A}_k}{\overline{A}_1 + \overline{A}_2 + \ldots + \overline{A}_{n-1}} \geq \frac{\overline{B}_1 + \overline{B}_2 + \cdots + \overline{B}_k}{\overline{B}_1 + \overline{B}_2 + \ldots + \overline{B}_{n-1}}. \tag{1.8}$$

An interpretation of this condition (1.8) is as follows. Let X^A be the renewal time of some renewal process with $\Pr\{X^A = i\} = a_i$, $i = 1, \ldots, n-1$, and $a_0 = 0$. Then the equilibrium renewal time of X^A, called X_e^A, will be distributed by

$$\Pr\{X_e^A \leq k\} = \frac{\overline{A}_1 + \overline{A}_2 + \cdots + \overline{A}_k}{\overline{A}_1 + \overline{A}_2 + \cdots + \overline{A}_{n-1}}.$$

Therefore, (1.8) means $X_e^A \leq_{st} X_e^B$. Theorem 1.5 combined with Corollary 1.6 can be restated for position independent policies as follows.

Theorem 1.10 *Given two position independent policies* **A** *and* **B** *such that* $X_e^A \leq_{st} X_e^B$, *then* $\{\pi_i^A\} \leq_{st} \{\pi_i^B\}$ *and* $C(\mathbf{p}; \mathbf{A}) \leq_{st} C(\mathbf{p}; \mathbf{B})$ *for* $\mathbf{p} = (cp, p, \ldots, p), c > 1$.

Proof. Direct application of Theorem 1.5, Corollary 1.6 and Lemma 1.9. □

Note that the condition that A_{ij} is decreasing in i in Theorem 1.5. becomes $\overline{A}_i$ is decreasing in i which is true by its definition. An immediate result of Theorem 1.10 is that moving i positions closer is better than moving $i+1$ positions closer. Formally,

Corollary 1.11 *The steady state cost under move-i-position policy is stochastically smaller than that under move-$(i+1)$-position policy.*

Proof. Direct application of Theorem 1.10. □

Tenembaum and Nemes [9] examine two spectra of policies. For each spectrum, they show that the policies are ordered by their expected steady state cost, having tranposition at one end of the spectrum with minimum expected steady state cost and move-to-front at the other with maximum expected steady state cost. It can be shown that this also results directly from Theorem 1.5 and Corollary 1.6, and not only are the policies ordered by their expected steady state cost but their steady state costs are also stochastically smaller or larger than each other.

The first is a spectrum of policies $\mathrm{POS}(k), k = 1, \ldots, n$ where the requested item found at position j is moved to position k if $j > k$, and it is moved one position closer to the front if $j \leq k$. We can write the matrices $\mathbf{A}$ and $\mathbf{B}$ representing policies $\mathrm{POS}(k+1)$ and $\mathrm{POS}(k)$ respectively as follows.

$$\mathbf{A} = \begin{bmatrix}
1 & & & & & & & & & \\
1 & 1 & & & & & & & & \\
0 & 1 & \ddots & & & & & & & \\
\vdots & 0 & \ddots & 1 & & & & & & \\
 & \vdots & & 1 & 1 & & & & & \\
 & & & 0 & 1 & 1 & & & & \\
 & & & \vdots & \vdots & 1 & \ddots & & & \\
 & & & & & \vdots & \ddots & 1 & & \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & 1 & 1 & \\
0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & 1 & 1
\end{bmatrix}$$

Col. (1) $\cdots$ $(k+1)$ $\cdots$ (n)

$$\mathbf{B} = \begin{bmatrix} 1 & & & & & & & & & \\ 1 & 1 & & & & & & & & \\ 0 & 1 & \ddots & & & & & & & \\ \vdots & 0 & \ddots & 1 & & & & & & \\ & \vdots & & 1 & 1 & & & & & \\ & & & 0 & 1 & 1 & & & & \\ & & & \vdots & \vdots & 1 & \ddots & & & \\ & & & & & \vdots & \ddots & 1 & & \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & 1 & 1 & \\ 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & 1 & 1 \end{bmatrix}$$

Col. (1) $\cdots$ (k) $\cdots$ (n)

The upper triangles of both matrices **A** and **B** consist of zeros. It can be easily checked that both policies **A** and **B** are in S as they satisfy (1.7) of Proposition 1.8. Moreover, all the conditions of Theorem 1.5 are also satisfied. We can then make a stronger statement that the steady state cost under POS$(k+1)$ is stochastically smaller than the steady state cost under POS(k).

The second is a spectrum of plicies SWITCH$(k), k = 1, \ldots, n$, where the requested item found at j is moved one position closer if $j > k$, and is moved to the first position if $j \leq k$. Al the conditions of Theorem 1.5 and (1.7) of Proposition 1.8 are satisfied by the following matrices **A** and **B** representing SWITCH(k) and SWITHC$(k+1)$ respectively.

Row

$$\mathbf{A} = \begin{bmatrix} 1 & & & & & & & & & \\ 1 & 1 & & & & & & & & \\ \vdots & 1 & \ddots & & & & & & & \\ & \vdots & \ddots & 1 & & & & & & \\ 1 & 1 & \ldots & 1 & 1 & & & & & \\ 0 & 0 & \ldots & 0 & 1 & 1 & & & & \\ \vdots & \vdots & & \vdots & 0 & 1 & \ddots & & & \\ & & & & \vdots & 0 & \ddots & 1 & & \\ & & & & & \vdots & & 1 & 1 & \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix} \begin{matrix} (1) \\ \vdots \\ (k) \\ \vdots \\ \vdots \\ (n) \end{matrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & & & & & & & & & \\ 1 & 1 & & & & & & & & \\ \vdots & 1 & \ddots & & & & & & & \\ & \vdots & \ddots & 1 & & & & & & \\ 1 & 1 & \dots & 1 & 1 & & & & & \\ 0 & 0 & \dots & 0 & 1 & 1 & & & & \\ \vdots & \vdots & & \vdots & 0 & 1 & \ddots & & & \\ & & & & \vdots & 0 & \ddots & 1 & & \\ & & & & & \vdots & & 1 & 1 & \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix} \begin{matrix} \text{Row} \\ (1) \\ \vdots \\ (k+1) \\ \vdots \\ \vdots \\ (n) \end{matrix}$$

The upper triangles of **A** and **B** also consist of zeros. Similarly, the steady state cost under SWITCH(k) is stochastically smaller than that under SWITCH($k+1$).

2 The Processor Problem

Let $C(\mathbf{p};\mathbf{A})$ now be the steady state cost and $Y_1(\mathbf{p};\mathbf{A})$ the steady state position of processor 1 of the processor problem with success probability vector $\mathbf{p}$ under policy **A**. When only processor 1 has a different success probability, the expected steady state cost, conditioning on the position of processor 1, can be written as

$$\begin{aligned} E[C(\mathbf{p};\mathbf{A})] &= \sum_{i=1}^{n} E\Big[C(\mathbf{p};\mathbf{A})|Y_1(\mathbf{p};\mathbf{A}) = i\Big]\pi_i \\ &= \Big[1 + q_1(1 + q + \cdots + q^{n-2})\Big]\pi_1 \\ &+ \sum_{i=2}^{n-1}\Big[(1 + q + \cdots + q^{i-1}) + q_1(q^{i-1} + q^i + \cdots + q^{n-2})\Big]\pi_i \\ &+ (1 + q + \cdots + q^{n-1})\pi_n \\ &= \frac{1 - q_1 q^{n-1}}{p} - \frac{p_1 - p}{p}\sum_{i=1}^{n} q^{i-1}\pi_i, \end{aligned} \tag{2.1}$$

where $\mathbf{p} = (p_1, p, \ldots, p)$, $q_1 = 1 - p_1$, $q = 1 - p$ and π_i is the steady state probability that processor 1 is at position i. From (2.1), since q^i is decreasing in i, if $p_1 > p$ and the position of procesor 1 under policy **A** is stochastically smaller than under policy **B**, the expected steady state cost under policy **A** will be smaller than the expected steady state cost under policy **B**. For the rest of the paper, we assume that $p_1 > p$.

2.1 Randomized Policy

Define the randomized policy **A** and its related randomized policy $\mathbf{A}^d$ in exactly the same way as in the list problem. Also let π_i^A be the steady state probability that processor 1 is at position i under policy **A**. Define $K_i^A = \pi_i^A/\pi_n^A$. Lemma 2.1 to Lemma 2.4 below show the relationships between $\{\pi_i^A\}$ and $\{\pi_i^{dA}\}$ under the assumption that $(p_1^d, p_2^d, \ldots, p_{n-1}^d) = (p_1, p, \ldots, p)$.

Lemma 2.1 *Under policy* **A**, *for* $i = 2, \ldots, n$,

$$\pi_i^A/\pi_n^A = K_i^A = \pi_{i-1}^{dA}/\pi_{n-1}^{dA}.$$

Proof. Similar to Lemma 1.1, the Lemma is proved by using the column $r+1$ of the transition matrix, which is given by

$$\begin{bmatrix} \cdots & 0 & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & 0 & \cdots \\ \cdots & q^{r-1}q_1p\sum_{i=r+1}^{n} q^{i-r-1}\sum_{j=1}^{r} a_{ij} & \cdots \\ \cdots & 1-q^r+q^rp_1a_{r+1,r+1}+q^rq_1p\left(\sum_{i=r+2}^{n} q^{i-r-2}\sum_{j=r+2}^{i} a_{ij}\right)+q^{n-1}q_1 & \cdots \\ \cdots & q^{r+1}p_1a_{r+2,r+1} & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & q^{r+1}p_1a_{n,r+1} & \cdots \end{bmatrix}$$

□

Lemma 2.2 *Under policy* **A**, *for* $i = 2, 3, \ldots, n$,

$$\pi_i^A = (1-\pi_1^A)\pi_{i-1}^{dA}.$$

Proof. Same as Lemma 1.2. □

Lemma 2.3 *Under policy* **A**,

$$\pi_1^A = \frac{p_1q}{q_1p}\left(\frac{a_{21}\pi_2^A + qa_{31}\pi_3^A + \cdots + q^{n-2}a_{n1}\pi_n^A}{a_{21} + qa_{31} + \cdots + q^{n-2}a_{n1}}\right).$$

Proof. Similar to Lemma 1.3, the Lemma is proved by using the first column of the transition matrix, which is given by

$$\begin{bmatrix} p_1 + q_1p\sum_{i=2}^{n} q^{i-2}\sum_{j=2}^{i} a_{ij} + q^{n-1}q_1 & \cdots \\ qp_1a_{21} & \cdots \\ q^2p_1a_{31} & \cdots \\ \vdots & \cdots \\ q^{n-1}p_1a_{n1} & \cdots \end{bmatrix}$$

□

Lemma 2.4 *Under policy* **A**, *for* $j = 1, 2, \ldots, n-1$,

$$K_j^A = \frac{p_1 q}{q_1 p}\left(\frac{A_{j+1,j}K_{j+1}^A + qA_{j+2,j}K_{j+2}^A + \cdots + q^{n-j-1}A_{nj}K_n^A}{A_{j+1,j} + qA_{j+2,j} + \cdots + q^{n-j-1}A_{nj}}\right).$$

Proof. Same as Lemma 1.4. □

2.2 Comparison of the Steady State Costs and Probabilities of Two Problems under Two Different Policies

We can now state a result similar to Theorem 1.5 that compares the steady state probability $\{\pi_i\}$ under two different policies. As in the list problem, let S be the set of policies under which the resulting probability distribution $\{\pi_i\}$ is decreasing in i when $p_1 > p$ and increasing in i otherwise.

Theorem 2.5 *Let* **A** *and* **B** *be two policies such that, for* $j = 1, 2, \ldots, n-1$, $k = j+1, \ldots, n$,

$$\frac{A_{j+1,j} + qA_{j+2,j} + \cdots + q^{k-j-1}A_{kj}}{A_{j+1,j} + qA_{j+2,j} + \cdots + q^{n-j-1}A_{nj}} \geq \frac{B_{j+1,j} + qB_{j+2,j} + \cdots + q^{k-j-1}B_{kj}}{B_{j+1,j} + qB_{j+2,j} + \cdots + q^{n-j-1}B_{nj}}, \tag{2.2}$$

and at least one of these two conditions holds:

(a) $\mathbf{A} \in S$ *and* B_{ij} *is decreasing in* i *for all* $j = 1, \ldots, n$,
(b) $\mathbf{B} \in S$ *and* A_{ij} *is decreasing in* i *for all* $j = 1, \ldots, n$.

Then $\{\pi_i^A\} \leq_{st} \{\pi_i^B\}$ *for any* $\mathbf{p} = (p_1, p, \ldots, p), p_1 > p$.

Proof. Same as Theorem 1.5 because if A_{ij} is decreasing in i for all j then so is $q^{i-j-1}A_{ij}$. □

It should be noted that (1.6) and (2.2) are not equivalent when A_{ij} and B_{ij} are decreasing in i for all j, even though (2.2) gives (1.6) when $q = 1$. A simple counterexample can be constructed as follows. Suppose (1.6) is true. Let $j = 1$ and $A_{21} + A_{31} + \cdots + A_{n1} = B_{21} + B_{31} + \cdots + B_{n1}$, with $A_{21} = B_{21}$. So by (1.6), $(A_{21}, A_{31}, \ldots, A_{n1})$ majorizes $(B_{21}, B_{31}, \ldots, B_{n1})$. With the fact that q^i is decreasing in i, we have

$$A_{21} + qA_{31} + \cdots + q^{n-2}A_{n1} \geq B_{21} + qB_{31} + \cdots + q^{n-2}B_{n1},$$

which means

$$\frac{A_{21}}{A_{21} + qA_{31} + \cdots + q^{n-2}A_{n1}} \leq \frac{B_{21}}{B_{21} + qB_{31} + \cdots + q^{n-2}B_{n1}}.$$

This contradicts (2.2) for $j = 1$ and $k = 2$.

A consequence of Theorem 2.5 is that the steady state cost under policy **A** is stochastically smaller than the steady state cost under policy **B**.

Corollary 2.6 *Under the conditions of Theorem 2.5,* $C(\mathbf{p}; \mathbf{A}) \leq_{st} C(\mathbf{p}; \mathbf{B})$.

Proof. Same as Corollary 1.6. □

By exactly the same reason as in Corollary 1.7, transposition has a steady state cost stochastically smaller than any randomized policy.

Corollary 2.7 *For any policy* $\mathbf{A}, C(\mathbf{p}; T) \leq_{st} C(\mathbf{p}; \mathbf{A})$.

Proof. Same as Corollary 1.7. □

A counterexample similar to that in Section 1.2 can be made to show that not every randomized policy is in S. A sufficient condition for a policy $\mathbf{A}$ to be in S turns out to be the same as in the list problem. That is, when $p_1 > p$, $\{\pi_i^A\}$ is decreasing in i when (2.3) below, which is (1.7) of Proposition 1.8, holds.

Proposition 2.8 *A policy* $\mathbf{A} \in S$ *if, for* $j = 1, \ldots, n-1$,

$$\frac{A_{j+1,j} + \cdots + A_{nj}}{A_{j+2,j+1} + \cdots + A_{n,j+1}} \leq \frac{A_{j+1,j} + \cdots + A_{n-1,j}}{A_{j+2,j+1} + \cdots + A_{n-1,j+1}} \leq \cdots \leq \frac{A_{j+1,j} + A_{j+2,j}}{A_{j+2,j+1}}. \tag{2.3}$$

Proof. By the same argument as in Proposition 1.8, $\mathbf{A} \in S$ if, for $k = j+3, \ldots, n$,

$$\frac{A_{j+1,j} + qA_{j+2,j} + \cdots + q^{k-j-1}A_{kj}}{A_{j+2,j+1} + qA_{j+3,j+1} + \cdots + q^{k-j-2}A_{k,j+1}}$$

$$\leq \frac{A_{j+1,j} + qA_{j+2,j} + \cdots + q^{k-j-2}A_{k-1,j}}{A_{j+2,j+1} + qA_{j+3,j+1} + \cdots + q^{k-j-3}A_{k-1,j+1}}. \tag{2.4}$$

It is then sufficient to show that (2.3) implies (2.4). By cross-multiplying and rearranging terms, (2.4) is equivalent to

$$\frac{qA_{kj}}{A_{j+1,j} + qA_{j+2,j} + \cdots + q^{k-j-2}A_{k-1,j}} \frac{A_{j+1,j} + \cdots + A_{k-1,j}}{A_{j+1,j} + \cdots + A_{k-1,j}}$$

$$\leq \frac{A_{k,j+1}}{A_{j+2,j+1} + qA_{j+3,j+1} + \cdots + q^{k-j-3}A_{k-1,j+1}} \frac{A_{j+2,j+1} + \cdots + A_{k-1,j+1}}{A_{j+2,j+1} + \cdots + A_{k-1,j+1}}. \tag{2.5}$$

Now,

$$\frac{A_{kj}}{A_{j+1,j} + \cdots + A_{k-1,j}} \leq \frac{A_{k,j+1}}{A_{j+2,j+1} + \cdots + A_{k-1,j+1}}$$

$$\Leftrightarrow \frac{A_{j+1,j} + \cdots + A_{kj}}{A_{j+2,j+1} + \cdots + A_{k,j+1}} \leq \frac{A_{j+1,j} + \cdots + A_{k-1,j}}{A_{j+2,j+1} + \cdots + A_{k-1,j+1}}, \tag{2.6}$$

where the inequality on the right hand side of the equivalence is given by (2.3). Also from (2.3), for $m \leq k-1$,

$$\frac{A_{j+1,j} + \cdots + A_{mj}}{A_{j+1,j} + \cdots + A_{k-1,j}} \geq \frac{A_{j+2,j+1} + \cdots + A_{m,j+1}}{A_{j+2,j+1} + \cdots + A_{k-1,j+1}},$$

and because q^i is decreasing in i we have

$$\begin{aligned} &\frac{A_{j+1,j} + qA_{j+2,j} + \cdots + q^{k-j-2}A_{k-1,j}}{A_{j+1,j} + A_{j+2,j} + \cdots + A_{k-1,j}} \\ \geq\ &\frac{qA_{j+2,j+1} + q^2A_{j+3,j+1} + \cdots + q^{k-j-2}A_{k,j+1}}{A_{j+2,j+1} + A_{j+3,j+1} + \cdots + A_{k,j+1}}. \end{aligned} \tag{2.7}$$

Then (2.5) follows from (2.6) and (2.7). □

Thus for the processor problem, by the same argument as in Lemma 1.9, any position independent policy is also in S. Formally,

Lemma 2.9 *Let* **A** *be a position independent policy that moves the succesful processor I positions with probability* a_i, $\sum_{i=0}^{n-1} a_i = 1$. *Then* $\mathbf{A} \in S$.

Proof. Same as Lemma 1.9. □

We can then restate Theorem 2.5 combined with Corollary 2.6 for position independent policies as follows.

Theorem 2.10 *Given two position independent policies* **A** *and* **B** *such that, for* $k = 1, \ldots, n-1$,

$$\frac{\overline{A}_1 + q\overline{A}_2 + \cdots + q^{k-1}\overline{A}_k}{\overline{A}_1 + q\overline{A}_2 + \cdots + q^{n-2}\overline{A}_{n-1}} \geq \frac{\overline{B}_1 + q\overline{B}_2 + \cdots + q^{k-1}\overline{B}_k}{\overline{B}_1 + q\overline{B}_2 + \cdots + q^{n-2}\overline{B}_{n-1}}, \tag{2.8}$$

then $\{\pi_i^A\} \leq_{st} \{\pi_i^B\}$ *and* $C(\mathbf{p}; \mathbf{A}) \leq_{st} C(\mathbf{p}; \mathbf{B})$ *for any* $\mathbf{p} = (p_1, p, \ldots, p), p_1 > p$.

Proof. Direct application of Theorem 2.5 Corollary 2.6 and Lemma 2.9. □

There is no obvious interpretation of (2.8), unlike (1.8), as in the list problem. However, (2.8) yields the same monotonicity result as in the list problem that move-i-position has a steady state cost stochastically smaller than move-(i + 1)-position. Let **A** and **B** represent the move-i-position and move-(i + 1)-position policies respectively. Then,

$$\overline{A}_1 = \overline{A}_2 = \cdots = \overline{A}_i = 1, \overline{A}_{i+1} = \overline{A}_{i+2} = \cdots = \overline{A}_{n-1} = 0$$

$$\overline{B}_1 = \overline{B}_2 = \cdots = \overline{B}_{i+1} = 1, \overline{B}_{i+2} = \overline{B}_{i+3} = \cdots = \overline{B}_{n-1} = 0.$$

Therefore, for $k = 1, \ldots, n-1$,

$$\begin{aligned} \frac{\overline{A}_1 + q\overline{A}_2 + \cdots + q^{k-1}\overline{A}_k}{\overline{A}_1 + q\overline{A}_2 + \cdots + q^{n-2}\overline{A}_{n-1}} &= \frac{1 + q + \cdots + q^{k-1}}{1 + q + \cdots + q^{i-1}} \\ &\geq \frac{1 + q + \cdots + q^{k-1}}{1 + q + \cdots + q^{i}} \\ &= \frac{\overline{B}_1 + q\overline{B}_2 + \cdots + q^{k-1}\overline{B}_k}{\overline{B}_1 + q\overline{B}_2 + \cdots + q^{n-2}\overline{B}_{n-1}}. \end{aligned} \tag{2.9}$$

We have proved the following Corollary.

Corollary 2.11 *The steady state cost under the move-i-position policy is stochastically smaller than that under the move-$(i+1)$-position policy.*

Proof. By (2.9) and Theorem 2.10. □

By Theorem 2.5, it also holds, as in the case of the list problem shown in Section 1.2, that the policies in the two spectra of Tenenbaum and Nemes [9] are ordered such that the policies in each spectrum have steady state costs stochastically smaller or larger than each other.

Acknowledgements. I would like to thank my advisor Prof. Sheldon M. Ross, Anandhamahidol Foundation and Thammasat University.

References

[1] Anderson, E.J., Nash, P., and Weber, R.R. A counterexample to a conjecture on optimal list ordering. *Journal of Applied Probability,* Vol. 19, No. 3 (1982), 730-732.

[2] Gonnett, G.H., Munro, J.I., and Suwanda, H. Exegesis of self-organizing linear search. *SIAM Journal on Computing,* Vol. 10 (1981), No. 3, 613-637.

[3] Hendricks, W.J. The stationary distribution of an interesting Markov chain. *Journal of Applied Probability,* Vol. 9, No. 1 (1972), 231–23.

[4] Hendricks, W.J. An account of self-organizing systems. *SIAM Journal on Computing,* Vol. 5 (1976), No. 4, 715-723.

[5] Hester, J.H., and Hirschberg, D.S. Self-organizing Linear Search. *Computing Surveys,* Vol. 17, No. 3 (1985), 295-311.

[6] Kan, Y.C., and Ross, S.M. Optimal list order under partial memory constraints. *Journal of Applied Probability,* Vol. 17, No. 4 (1980), 1004-1015.

[7] Phelps, R.I., and Thomas, L.C. On optimal performance in self-organizing paging algorithms. *Acta Cybernetica,* Vol. 5, No. 1 (1980), 88-85.

[8] Ross, S.M. Processor reordering rules. *Probability in the Engineering and Information Science,* Vol. 4, No. 2 (1990), 181-186.

[9] Tenenbaum, A., and Nemes, R.M. Two spectra of self-organizing sequential search algorithms. *SIAM Journal on Computing,* Vol. 11, No. 3 (1982), 557-566.

[10] Topkis, D.M. Reordering heuristics for routing in communication networks. *Journal of Applied Probability,* Vol. 23, No. 1 (1986), 130-143.

Received October 21, 1991

Acta Cybernetica, Vol. 10, Nr. 4, Szeged, 1992

Computing Maximum Valued Regions

G. J. Woeginger*†

Abstract

We consider the problem of finding optimum connected configurations in the plane and in undirected graphs. First, we show that a special case concerning rectilinear grids in the plane and arising in oil business is NP-complete, and we present a fast approximation algorithm for it. Secondly, we identify a number of polynomial time solvable special cases for the corresponding problem in graphs. The special cases include trees, interval graphs, cographs and split graphs.

1 Introduction

Problem statement and applications. In this paper, we deal with the MAXIMUM VALUED REGION problem (MVR, for short) which is defined as follows. We are given a subdivision of a rectangle into equisized squares. Every single square has some (known) positive value. The problem is to find for a given number k a connected subregion of the rectangle that consists of exactly k squares and that has the maximum overall value under these conditions.

Practical applications of MVR arise e.g. in the context of oil business, cf. Hamacher, Joernsten and Maffioli [6]. Suppose a company is searching for oil at many places of some large area and assigns *values* to the pieces of land according to the results of these trial prospects. The places form some regular (rectilinear) pattern as described above. Afterwards, the company will buy the 'best' k landpieces; assuming unit prices for the land we exactly arrive at MVR.

A related graph problem. The corresponding problem in vertex-valued graphs is to find a connected subgraph on k vertices with maximum overall value. We call this graph problem the *Maximum Valued Subtree* problem, MVS for short. Problem MVS is known to be NP-complete for arbitrary graphs (see [6]). It is easy to see that MVS restricted to *gridgraphs* becomes MVR.

Known results. Hamacher et al. [6] introduced the problem MVS and proved it to be NP-complete for arbitrary graphs. They also developped a branch-and-bound scheme for MVS, and gave an integer program formulation. As a main open problem they asked whether the restriction of MVS to gridgraphs can be solved in polynomial time. Maffioli [8] derived a polynomial time algorithm for solving MVS in trees.

*TU Graz, Institut für Theoretische Informatik, Klosterwiesgasse 32/II, A-8010 Graz, Austria. Electronic mail: gwoegi@igi.tu-graz.ac.at

†This research was supported by the Christian Doppler Laboratorium für Diskrete Optimierung.

Our results. We prove that MVR (and hence the restriction of MVS to gridgraphs) is NP-complete and we give a polynomial time approximation algorithm with worst case guarantee $O(\sqrt{k})$ (i.e. the approximation algorithm always outputs a solution with value at least the optimum value divided by $c\sqrt{k}$).

For the graph problem MVS, we will identify several polynomial time solvable subcases, e.g. MVS in trees, interval graphs and cographs. It turns out that MVS and the famous STEINER TREE problem are closely related in the following sense: The investigated restrictions to the various 'famous' graph classes (as described in Johnson [7]) are either NP-complete for *both* problems or polynomial time solvable for *both* problems.

Organization of the paper. In Section 2, we give the NP-completeness proof for MVR, and in Section 3 our approximation algorithm is described and analyzed. Section 4 deals with treelike graph classes for which MVS is polynomial time solvable by a dynamic programming approach. Section 5 summarizes some other results on special graph classes (interval graphs, cographs and split graphs). Section 6 contains the discussion.

2 NP-completeness of the Region Problem

To give a precise presentation of the problem and our method, we will need the following definitions. A *gridpoint* in the Euclidean plane is a point with both coordinates integer. Two gridpoints are called *adjacent* iff they are at distance one from each other. This adjacency relation induces an infinite graph on the gridpoints. A *region* is a set of gridpoints that is connected in this infinite graph.

MAXIMUM VALUED REGION PROBLEM (MVR)
Input. A rectangular region $R = [1, .., n] \times [1, .., n]$; with sidelength n; a value function $c : R \rightarrow \mathbb{Z}^+$; a positive integer k; an integer bound C.
Question. Does there exist a region $R' \subseteq R$ of exactly k points with total value $c(R') \geq C$?

We will show that the NP-complete *planar Steiner Tree problem* (cf. Garey and Johnson [3]) is polynomial time reducible to our problem MVR.

STEINER TREE IN PLANAR GRAPHS (PST)
Input. A planar graph $G = (V, E)$; a weight $w : E \rightarrow \mathbb{Z}$; a subset $X \subseteq V$; an integer bound $W \in \mathbb{Z}$.
Question. Does there exist a Steiner Tree $T = (V_T, E_T)$ of G for X (i.e. does there exist a subtree T of G with $X \subseteq V_S$) such that $w(E_T) \leq W$?

We start with an instance of PST and construct from this an instance of MVR that is solvable if and only if PST is solvable. To simplify the presentation, we will also use negative values for points in problem MVR. Since exactly k points have to be chosen, adding a large positive constant to all values yields an equivalent problem with positive values.

In a first step, we compute a *rectilinear planar layout* of the graph G. Such a layout maps the vertices of G to (pairwise disjoint) horizontal line segments and maps the edges of G to (pairwise disjoint) vertical line segments, with all endpoints

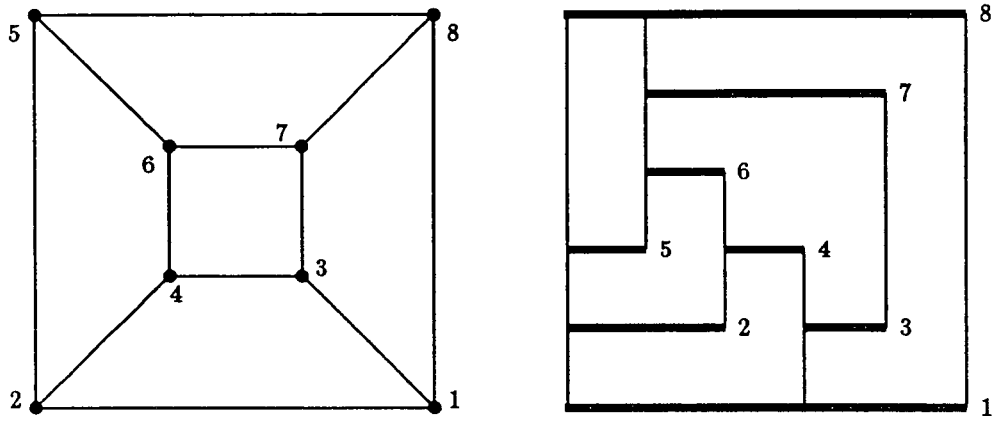

Figure 1: A planar graph and its rectilinear planar layout.

of segments at positive integer coordinates. Two horizontal vertex-segments are connected by a vertical edge-segment, if and only if the corresponding vertices are adjacent in the graph. Figure 1 shows a drawing of a planar graph together with its rectilinear planar layout.

Rosenstiehl and Tarjan [9] show how to compute a rectilinear planar layout for planar graphs with n vertices in $O(n)$ time. The height of their layout is at most $|V|$, and the width is at most $2|V|-4$. Most important, one can choose an arbitrary vertex to become the bottom horizontal vertex-segment of the layout. We choose some vertex x in X to become the bottom segment.

In the second step, we stretch the rectilinear planar layout in horizontal and vertical directions by a factor of two, i.e. we multiply the coordinates of all endpoints by 2. This ensures that points on distinct segments are at distance at least two, unless they correspond to a vertex-edge incidence.

In the third and last step, we finally transform the layout into a weighted region for problem MVR. We distinguish five types of gridpoints: vertex-points, edge-points, link-points, dummy-points and fill-points. The vertex-points, edge-points and link-points together cover exactly all gridpoints on the line segments of the stretched rectilinear planar layout.

- For each vertex v in X (the set that has to be spanned by the Steiner Tree), we choose an arbitrary gridpoint on the horizontal line segment corresponding to v and make it a *vertex-point* $G(v)$. The value c of every point $G(v)$ is set to $M := \sum_{e \in E} w(e) + 1$.

- For each edge e in E, we choose an arbitrary gridpoint on the vertical line segment corresponding to e and make it an *edge-point* $G(e)$. The value $c(G(e))$ equals the weight $-w(e)$.

- All points lying on line segments of the stretched rectilinear planar layout that are neither vertex-points nor edge-points become link-points and receive a value of 0.

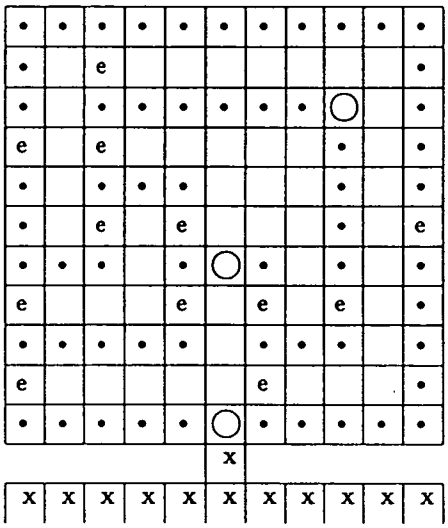

Figure 2: The upper part of the constructed region instance.

Now let n_v, n_e and n_ℓ denote the number of vertex-, edge- and link-points, respectively. Set $k = n_v + n_e + n_\ell$ and observe that $k \in O(|V|^2)$.

- We create a connected region of k points *just below* the layout, separated from the layout by a single row of unused gridpoints (in other words, the topmost points in this new region are at distance two from the lower border of the layout). Moreover, we connect this region by a single gridpoint in this unused row to the vertex-point corresponding to vertex $x \in X$. The gridpoints in this new region and this single gridpoint constitute the set of dummy-points. All of them have value 0.

- Finally, we enclose the vertex-, edge-, link- and dummy-points by a rectangle (obviously, the sidelength of this rectangle is polynomial in $|V|$). All points in this rectangle to which we did not assign a value till this moment are the fill points; they have value $-|X| \cdot M - 1$.

An illustration for this construction is given in Figure 2, where the graph depicted in Figure 1 is transformed in its corresponding region. The vertices in X are 1, 4 and 7 and the corresponding vertex-points in the region are marked by a "○". The edge-points are marked by an "e", and the link-points by a "•". Empty space corresponds to fill points. For reasons of readability and space, we did not show all dummy-points (marked by an "x") that lie below the bottom "○".

We claim that the constructed instance of MVR has a solution with value at least $C = |X| \cdot M - W$ if and only if there exists a Steiner Tree of G for X with weight at most W. W.l.o.g. we assume $W \leq \sum_{e \in E} w(e) = M - 1$, as otherwise the PST is trivially solvable.

(only if) Assume there exists a region R' with value at least $|X| \cdot M - W$. Since the only points with positive value are exactly the $|X|$ vertex-points with value M

and since $W \leq M - 1$ holds, R' must contain all the vertex-points. Since all fill-points have value $-|X| \cdot M - 1$, no fill-point appears in R'.

Obviously, the dummy-points (outside of the layout!) cannot help in connecting the vertex-points. Hence, the connections must result from the edge- and from the link-points. Vertex-points on two distinct horizontal segments can be connected to each other only via points on the vertical edge-segments. Using link-points (with value 0) is no problem, but in the middle of each edge there sits an edge-point, substracting $w(e)$ from the value of region R'. In order to connect all vertex-points while substracting at most W from the total value $|X| \cdot M$ of all vertex-points, we must find a connected configuration that has edge-weight at most W and that contains X. This exactly yields the claimed Steiner Tree.

(if) Now assume we are given a Steiner Tree with edge-weight at most W. We start with putting into the region R' all gridpoints on line segments corresponding to edges and vertices used in the Steiner Tree. Hence, it contains all vertex-points (with total value $|X| \cdot M$), some edge-points (with total value at most W) and a number of link-points with zero value. By the definition of k, the overall number of these points is at most k. To get a region with exactly k points, we add an appropriate number of dummy-points to R' such that R' remains connected. As all dummy points have zero value, the total value of the constructed R' is at least $|X| \cdot M - W$.

Summarizing, we have proven the following theorem.

Theorem 2.1 *Problem* MVR *is NP-complete.* □

3 A Heuristic for MVR

In this section we analyze the following fast and simple heuristic for the region problem MVR (for convenience we assume throughout this section that $k = \alpha^2$ is a square number).

> Take the highest value axes-parallel quadratic region Q^* with sidelength $\sqrt{k}$.

Clearly, this quadratic region can be found in $O(kn^2)$ time and a slightly more sophisticated implementation runs in $O(\sqrt{k}n^2)$ time. Our main interest is to determine the worst case quality of Q^* compared to the optimum region R^*.

Let $k = \alpha^2$. Consider a *staircase*, consisting of $\alpha/2$ vertical and $\alpha/2$ horizontal line segments where each line segment contains exactly α gridpoints. All gridpoints on the staircase receive value one, all other gridpoints receive value zero. Then the optimum region R^* has value k, whereas no square with sidelength α can cover more than 2α gridpoints on the staircase. Hence, for this configuration our heuristic is a factor of $\Omega(\sqrt{k})$ away from the optimum. It is easy to see that the staircase is also a bad configuration for the more general heuristic where we do not only consider axes-parallel squares but also arbitrary (not necessarily axes-parallel) rectangles.

Surprisingly, $O(\sqrt{k})$ is also the worst that can happen as will be shown in the remaining part of this section. We cover the optimum region R^* by an orthogonal grid with gridlength α and vertices that are integer points shifted by the vector $(1/2, 1/2)$. This grid is called the A-grid and it partitions the plane into A-cells.

Lemma 3.1 *At most 10α of the A-cells contain a point of R^*.*

Proof. Denote by V_R the set of all A-cells that contain at least one point of R^*. Two A-cells A_1 and A_2 in V_R are called adjacent iff A_i contains a gridpoint $p_i \in R^*$, $1 \leq i \leq 2$, such that p_1 and p_2 are at distance one. Let $T = (V_R, E_R)$ be an arbitrary spanning tree of the graph induced by this adjacency relation. We root T at an arbitrary leaf of T; this defines fathers and sons, and by the choice of the root no vertex has more than three sons.

Then we repeat the following two steps over and over again until T contains less than 10 vertices. (The tree is processed in a bottom-up fashion from the leaves towards the root. Step 1 removes subtrees that are paths on four vertices, and Step 2 removes 'branching' subtrees).

(Step 1). Assume, T contains a vertex v whose only descendants form a path on three vertices v_1, v_2 and v_3 (such a subgraph is called a *type-1 subgraph*). Then the corresponding four A-cells contain at least α points of R^*. The only interesting case occurs when the A-cells corresponding to v, v_1, v_2 and v_3 form a 2×2 configuration. W.l.o.g. let v be the lower left A-cell in this configuration, and let v be connected to its father via its left side. Then this left side must be linked to the diametric A-cell v_2. A-cell v_2 is at distance α from the left side of v, and there are at least α vertices necessary to link them.

We iteratively remove all type-1 subgraphs from T.

(Step 2). Next, we consider some vertex w of degree at least two with no descandants of degree at least two (w must exist, unless Step 1 deleted all but three vertices; in this case we terminate).

Vertex w has at most three sons, and since T does not contain any type-1 configuration any more, each of its sons has at most two descendants. Summarizing, the maximal subgraph of T rooted at w contains at most 10 A-cells. On the other hand, the A-cell corresponding to w has outside connections on at least three of its four sides (one to its father, and at least two to its sons). Two of these outside connections must lie on opposite sides of the cell, and in order to link them to each other, the A-cell must contain at least α points of R^*.

We remove the maximal subgraph of T rooted at w from T and return to Step 1.

To finish the proof of the lemma, we observe that each removal operation in Step 1 and 2 removes (i) at most 10 A-cells and (ii) at least α points of R^*. Because of (ii), at most k/α removal operations are performed and because of (i), T contains at most $10k/\alpha = 10\alpha$ A-cells. □

Theorem 3.2 *There exist constants $c_1 > c_2 > 0$ such that the heuristic detects for all instances a region Q^* whose value is at least the value of R^* divided by $c_1\sqrt{k}$, and there exist instances for which the value of Q^* is at most the value of R^* divided by $c_2\sqrt{k}$.*

Proof. We prove the statement for $c_1 = 10$ and $c_2 = 1/2$.

Applying Lemma 3.1 and an averaging argument, we see that there exists an A-cell A_0 such that the points in $R^* \cap A_0$ have overall value at least the optimum value divided by $10\sqrt{k}$. Since all other gridpoints in A_0 have nonnegative value, the value of Q^* is at least the value of R^* divided by $10\sqrt{k}$.

The lower bound follows from the staircase configuration described at the beginning of this section. □

Remark. The factor 10 in the statement of Lemma 3.1 is not the smallest possible. A more elaborate argument decreases the factor down to 4. For our purposes, any constant factor suffices.

4 Results for Trees

In this section, we consider the following problem corresponding to MVR in graphs.

> MAXIMUM VALUED SUBTREE PROBLEM (MVS)
> **Input.** A graph $G = (V, E)$; a value function $c : V \to \mathbb{N}$; a positive integer k and an integer bound C.
> **Problem.** Does there exist a k-subtree T of G with total value at least C ?

For general graphs, problem MVS is NP-complete since MVR is a special case of MVS. We will present a polynomial time result for trees. As usual, we assume that the input graph G is given by its *adjacency list*, i.e. for each vertex $v \in V$ we have a list of its neighbors in G. The number of vertices in G will always be denoted by n. A tree on k vertices will also be called a *k-tree* or a *k-subtree*.

Subsection 4.1 analyzes a related matrix problem and Subsection 4.2 gives an $O(nk^2)$ algorithm for MVS on trees, The algorithms are based on Dynamic Programming approaches. Maffioli [8] derived another (more complicated) polynomial time algorithm for MVS in trees with the same running time as our solution.

4.1 A Matrix Problem

In this subsection we will analyze a matrix problem that is closely related to the MVS-problem. Let M be a matrix with nonnegative integer entries that consists of d rows and k columns. We define that the entry in the i-th row and j-th column has *value* $M_{i,j}$ and *weight* j. For $0 \leq j \leq k$, we denote by $\text{MAXVAL}(M, j)$ the maximum value for which there exists a subset S_j of the entries in M fulfilling the following conditions.

- S_j contains at most one entry from every single row in M,
- the overall weight of S_j equals j, and
- the overall value of S_j is $\text{MAXVAL}_M(j)$.

Lemma 4.1 *For a $d \times k$ matrix M, all numbers* $\text{MAXVAL}(M, 0), \ldots, \text{MAXVAL}(M, k)$ *can be computed in overall time $O(k^2 d)$.*

Proof. We apply the *Divide and Conquer* paradigm to solve the problem by the following recursive procedure.

(1) We divide matrix M by a horizontal line into an upper and into a lower submatrix of equal size. We call these two submatrices U and L.

(2) We recursively calculate all numbers $\text{MAXVAL}(U, *)$ and $\text{MAXVAL}(L, *)$.

(3) We determine MAXVAL$(M, *)$ from MAXVAL$(U, *)$ and MAXVAL$(L, *)$ according to the formula

$$\text{MAXVAL}(M, j) = \max_{0 \le i \le j} \text{MAXVAL}(U, j-i) + \text{MAXVAL}(L, i)$$

for all j, $0 \le j \le k$.

The correctness of the algorithm is obvious. Since the Divide-Step (1) takes only constant time and the Merge-Step (3) is done with at most k^2 operations, the time complexity $T(d, k)$ fulfills the inequality

$$T(d, k) \le 2T(d/2, k) + k^2.$$

Standard calculations yield $T(d, k) \le dT(1, k) + dk^2$, and consequently the time complexity is at most $O(dk^2)$. □

4.2 Trees

Now let the tree $G = (V, E)$ constitute an instance of MVS with $n = |V| = |E| + 1$. We root G at an arbitrary vertex r. This assigns to every vertex v (with exception of the root r) a unique father $f(v)$. With every vertex $v \in V$, we associate the maximal subtree rooted at v. Let $v_1, v_2, \ldots, v_n$ be an enumeration of the vertices in V such that each v comes before its father $f(v)$. Such an enumeration can easily be found in $O(n)$ time.

We introduce a two-dimensional integer array AR$[i, j]$ with $n(k+1)$ entries. The rows are indexed by the vertices v_i in the above enumeration, and the columns are indexed by the numbers from 0 to k.

> The meaning of "AR$[i, j] = w$" is that "the maximum value j-subtree of $T(v_i)$ that *also contains its root* v_i, has value w".

Lemma 4.2 *The values of all entries in* AR *can be calculated in* $O(k^2 n)$ *time.*

Proof. We consecutively calculate all rows of AR, starting with the row corresponding to v_1 and ending with the row corresponding to $v_n = r$.

If $T(v_i)$ consists of the single vertex v_i, we set AR$[i, 0] = 0$, AR$[i, 1] = c(v_i)$ and all other entries in AR$[i, *]$ to $-\infty$.

If $T(v_i)$ consists of at least two vertices, we consider the sons $v_{m_1}, v_{m_2}, \ldots, v_{m_d}$ of v_i, where $d = \deg(v_i) - 1$. In order to compute AR$[i, j]$, we must find the optimum partitioning of the number $j - 1$ into d nonnegative numbers $j_1, \ldots, j_d$ that maximizes $\sum_{i=1}^{d}$ AR$[v_{m_i}, j_i]$. But this exactly amounts to solving the matrix problem treated in Section 1 on the submatrix M of AR$[*, *]$ generated by the rows corresponding to the vertices $v_{m_1}, v_{m_2}, \ldots, v_{m_d}$. According to Lemma 4.1, this problem can be solved in $O(k^2 d)$ time. Finally, we add to each of the k resulting numbers the value $c(v_i)$ of the root of this subtree.

To get the overall time complexity for computing AR$[*, *]$, we have to sum up the $k^2(\deg(v_i) - 1)$ steps for every v_i with at least one son plus the k steps for every v_i without a son. This is clearly dominated by $k^2 \sum_i \deg(v_i) \in O(k^2 n)$. □

Theorem 4.3 *For trees, the problem* MVS *can be solved in* $O(k^2n)$ *time and* $O(kn)$ *space.*

Proof. Assume that the maximum value k-subtree T is spanned by vertices $W = \{w_1, \ldots, w_k\}$ and let w denote the unique vertex in W whose father is not in W. Then the value of T equals $\text{AR}[w, k]$. Conversely, each entry in $\text{AR}[*, j]$ corresponds to a j-subtree.

Hence, the maximum number in the k-th column of $\text{AR}[*, *]$ gives the value of the maximum value k-subtree. The time complexity follows from the preceding lemma, the space complexity is determined by the size of AR. □

Remark. We only showed how to find the *value* of the maximum value k-subtree. If we also want to find the corresponding *k-subtree*, we have to store for each entry in $\text{AR}[*, *]$ its 'history' consisting of at most $k - 1$ predecessor entries as used in the dynamic program. This increases the time and the space complexity both by a factor of k.

5 Other Graph Families

This section deals with interval graphs, cographs and split graphs. We derive polynomial time results for the former two graph families and an NP-completeness proof for the latter family.

5.1 Interval Graphs

The vertices of an *interval graph* $G = (V, E)$ can be represented by intervals on the real line in such a way that two intervals intersect if and only if the corresponding vertices are adjacent. Most NP-complete graph problems become polynomial time solvable when restricted to interval graphs, cf. [5,7].

W.l.o.g. we may assume that intervals corresponding to distinct vertices have distinct endpoints. To find the MVS of a vertex-valued interval graph, we use the following decomposition of a *connected* interval graph G: The interval with the rightmost right endpoint is called the *head* of G. In general, there will be several intervals covering the left endpoint of the head. Among those intervals we choose that one with leftmost left endpoint, and we call it the *neck* of G; its endpoints are denoted by n_l and n_r. The remaining intervals either belong to the *body* of G (if their right endpoint lies to the left of n_r) or to the *hairs* of G (if their right endpoint lies to the right of n_r). Intuitively speaking, the body intervals are connected to the head via the neck. The hair intervals are directly connected to the head (their left endpoints are to the right of n_l, and their right endpoints are covered by the head).

Now sort the intervals from left to right according to their right endpoint and call the resulting sequence $I_1, \ldots, I_n$. We construct a twodimensional array $\text{AR}[1 \ldots |V|, 1 \ldots k]$ such that the maximum valued subtree with head I_j and consisting of k' vertices $(1 \leq k' \leq k)$ has value $\text{AR}[j, k']$. We compute all values in $\text{AR}[*, *]$, starting with level $\text{AR}[1, *]$ and going up to level $\text{AR}[n, *]$. The initialization steps are trivial, hence we only show how to compute $\text{AR}[j, k']$ for some fixed $j > k'$.

By definition, I_j constitutes the head of the optimum subgraph $G'(j, k')$ we are looking for. There are at most $n-1$ possibilities for the neck of $G'(j, k')$. There are at most $O(k^2)$ pairs (k_1, k_2) with sum $k_1 + k_2 + 2 = k'$, where k_1 denotes the number

of hairs and k_2 denotes the number of body–vertices. For fixed head and neck and for fixed numbers k_1 and k_2, the value of the optimum $G'(j,k')$ can be found in the following way. The body is the maximum valued connected subgraph on k_2+1 vertices and with our neck as new head; its value has already been computed and we find it in constant time. The hairs are the k_1 most precious intervals with left endpoints to the right of the left endpoint of the neck, and with right endpoints covered by the head. We claim that the optimum value for the k_1 hairs can be calculated in constant time with $O(nk)$ preprocessing for every head.

For a fixed head h, we enumerate all intersecting intervals sorted by their left endpoints from left to right in $O(n)$ time (in a preprocessing step, we sort all intervals by their left endpoints; if we deal with a fixed head, we run thru this list and select all intersecting intervals). We run through this enumeration from right to left and always store the k most precious values in a balanced tree: if the current interval has a value larger than the minimum in the tree, we remove the minimum and insert the value of the current interval (in case the tree has less than k vertices, we just insert the new value). Hence, we know in every single step the $1 \leq k' \leq k$ largest values and can compute their sum in $O(k)$ time.

Theorem 5.1 *For interval graphs, the problem* MVS *can be solved in* $O(k^2n^2)$ *time and* $O(kn)$ *space.*

Proof. The approach described above takes $O(k_1k_2n+kn)$ time for each of the n possible heads. Hence, the overall time is in $O(k^2n^2)$. The space requirements are dominated by the space of array AR. □

5.2 Cographs

In this section, we give a polynomial time algorithm for MVS in cographs.

Definition 5.2 *For* $r \geq 2$ *disjoint graphs* $G_i = (V_i, E_i)$ *with* $V_i \cap V_j = \emptyset$ *for* $i \neq j$, *the* union $\bigcup_{i=1}^{r} G_i$ *is defined as the graph* $(\bigcup_{i=1}^{r} V_i, \bigcup_{i=1}^{r} E_i)$. *Their* product $\times_{i=1}^{r} G_i$ *is obtained by first taking the union of the* r *graphs and then adding all edges* (v_i, v_j) *with* $v_i \in V_i$, $v_j \in V_j$ *and* $i \neq j$. □

Definition 5.3 *The class of* cographs *is the smallest set of graphs fulfilling the following rules.*

1. *The graph with one vertex and no edges is a cograph.*
2. *If* G_i, $1 \leq i \leq r$ *are cographs with pairwise disjoint vertex sets, then their union is a cograph.*
3. *If* G_i, $1 \leq i \leq r$ *are cographs with pairwise disjoint vertex sets, then their product is a cograph.* □

To each cograph $G = (V, E)$, we associate a corresponding rooted tree $T = (I, F)$, called the *cotree* of G and reflecting the above definition in the following way. Each non-leaf vertex in the tree is labeled either with $\cup$ (union-vertex) or $\times$ (product-vertex) and has two or more children. If two non-leaf vertices are connected by an edge, then they have different labels. Each vertex $x \in I$ of the cotree corresponds to a cograph $G_x = (V_x, E_x)$, and a leaf corresponds to a single-vertex graph. A union-vertex (product-vertex) corresponds to the union (product) of the cographs associated with the children of the vertex. Finally, the entire cograph is given by the cograph associated with the root $r \in I$ of the cotree. Corneil, Perl and Stewart [2] have shown that one can decide in linear time $O(|V| + |E|)$, whether a graph is a cograph, and build the corresponding cotree.

Theorem 5.4 *For a cograph $G = (V, E)$, the problem* MVS *can be solved in $O(k^2 n)$ time and $O(kn)$ space.*

Proof. We will compute two twodimensional arrays $\text{ARC}[x, k']$ and $\text{ARA}[x, k']$, where the rows correspond to the vertices x of the cotree and where $0 \leq k' \leq k$ holds. Once more, we start the computation of the array values at the leaves and go up to the root. $\text{ARC}[x, k']$ stores the largest possible value of any *connected* subgraph on k' vertices of the cograph G_x associated with x, and $\text{ARA}[x, k']$ stores the corresponding value for *arbitrary* (not necessarily connected) subgraphs.

The initialization is straight forward and we only show how to compute $\text{ARC}[x, *]$ and $\text{ARA}[x, *]$ for a non-leaf vertex x. The computation of $\text{ARA}[x, k']$ is easy: We simply take the k' most valuable vertices in the corresponding cograph. Applying e.g. the matrix algorithm from Subsection 4.1 this can be performed in $O(k^2 n)$ overall time for all vertices in the cotree. The computation of $\text{ARC}[x, k']$ is more envolved; we have to distinguish between union- and product-vertices x. For a union-vertex x, $\text{ARC}[x, k']$ equals the maximum of $\text{ARC}[s, k']$ over all sons s of x in the cotree (as a union operation cannot change connectivity properties of the graph). For a product-vertex x with sons $s_1, \ldots, s_p$, we perform two computations to find $\text{ARC}[x, k']$:

(i) We compute the maximum value of $\text{ARA}[s_1, k_1] + \ldots + \text{ARA}[s_p, k_p]$ over all p-tuples $(k_1, \ldots, k_p)$ with sum k' and at least two non-zero k_i (by applying the matrix algorithm). Since this maximum value results from at least two distinct sons of x, the corresponding graph is connected.

(ii) We compute $\max_i \text{ARC}[s_i, k']$. By the definition of $\text{ARC}[s_i, *]$, the corresponding graph is again connected.

Obviously, the maximum of the two values computed in (i) and (ii) yields $\text{ARC}[x, k']$.

The entry $\text{ARC}[r, k]$ for the root r of the cotree gives the desired value of the MVS. Since the cotree has $O(n)$ vertices, the claimed time and space complexity follows from the discussion in Subsection 4.1. □

5.3 Split Graphs

A graph $G = (V, E)$ is a *split graph*, if there is a partition of its vertices into an independent set I and in a clique C (and arbitrary edges between I and C), see Golumbic [4].

Theorem 5.5 *Problem* MVS *restricted to split graphs remains NP-complete.*

Proof. By reduction from the NP-complete SET-COVERING PROBLEM: Given a set $S = \{1, \ldots, p\}$ and subsets $A_1, \ldots, A_q \subset S$, the SET-COVERING PROBLEM consists in finding r subsets $A_{i_1}, \ldots, A_{i_r}$ with $\cup_{j=1}^{r} A_{i_j} = \{1, \ldots, p\}$. This problem is known to be NP-complete [3].

We construct a split graph on $p + q$ vertices that are labeled by some label in $\{1, \ldots, p, A_1, \ldots, A_q\}$. The vertices $A_1, \ldots, A_q$ form a clique, the vertices $1, \ldots, p$ form an independent set. We introduce an additional edge from ℓ to A_{i_j} iff $\ell \in A_{i_j}$ holds. All vertices $1, \ldots, p$ receive value 1, all other vertices receive value 0. Finally, we set $k = p + r$ and ask whether there exists a subtree with value at least p.

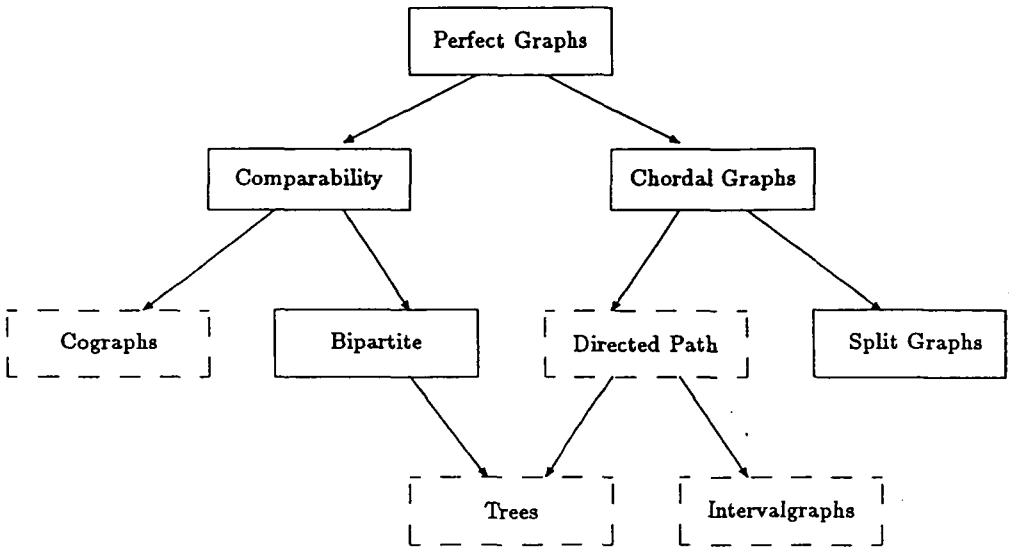

Figure 3: Containment relations for some of the treated graph classes.

In case such a tree exists, it uses all p vertices with value 1 and r vertices belonging to the clique must connect them; this yields the existence of a small set cover. In case a set cover with at most r subsets exists, we choose the corresponding r vertices in the clique and all p vertices not in the clique; clearly, the spanned graph is connected and of value p. □

Remark. We observe that there exists a simple approximation algorithm for MVS in split graphs with (tight) worst case guarantee 2: We simply take the $k/2$ most valuable vertices $v_1, \ldots, v_{k/2}$ and for every i some clique vertex c_i adjacent to v_i. Obviously, the resulting spanned graph is connected and its value is at least half of the optimum possible value.

6 Discussion

In this paper, we investigated the computational complexity of two closely related combinatorial problems, called MVR and MVS. The geometric problem MVR was shown to be NP-complete, and a polynomial time approximation algorithm was derived. The graph problem MVS is NP-complete for arbitrary graphs, but it can be solved efficiently on many well known special graph classes by applying Dynamic Programming techniques.

Figure 3 summarizes some of our results for MVS. Directed arcs represent containment of the lower graph class in the upper graph class. For classes with

a solid frame, MVS is NP-complete, and for classes with a dashed frame, MVS is polynomial time solvable (for exact definitions of all graph classes cf. Johnson [7]). Cographs, trees, interval graphs and split graphs were treated in this paper. The NP-completeness result for split graphs implies NP-completeness for chordal graphs and for perfect graphs. NP-completeness of MVS for bipartite graphs can be seen easily (by subdividing the edges of an arbitrary graph, assigning value zero to the new vertices and replacing k by $2k-1$), and this also yields the NP-completeness for comparability graphs. Finally, a polynomial time algorithm for directed path graphs can be derived by standard Dynamic Programming techniques (the method is analogous to that we applied to trees and cographs, and left to the ambitious reader as an exercise).

Moreover, for planar graphs we have proven the following results. MVS on grid-graphs (and consequently on arbitrary planar graphs) is NP-complete. However, the restriction to outerplanar and series-parallel graphs (these two classes are subsets of the partial 2-trees) can be solved in polynomial time. Bodlaender [1] derived an $O(k^2n)$ algorithm that solves MVS in partial K-trees, where K is not part of the input.

The most intriguing open problem is to construct polynomial time approximation algorithms for the geometric problem MVR with *constant* worst case guarantee (or prove that such algorithms do not exist).

References

[1] H.L.Bodlaender, private communication, 1992.

[2] D.G.Corneil, Y.Perl and L.K.Stewart, A linear recognition algorithm for cographs, *SIAM J. Comput.* **4**, 1985, 926–934.

[3] M.R.Garey and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[4] M.C.Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[5] U.I.Gupta, D.T.Lee and J.Y.-T.Leung, Efficient algorithms for interval graphs and circular–arc graphs, *Networks* **12**, 1982, 459–467.

[6] H.Hamacher, K.Joernsten and F.Maffioli, Weighted k-cardinality trees, Report 91.023, Dipartimento di Elettronica, Politecnico di Milano, 1991.

[7] D.S.Johnson, The NP-Completeness Column: an Ongoing Guide, *J. Algorithms* **6**, 1985, 434–451.

[8] F.Maffioli, Finding a best subtree of a tree, Report 91.041, Dipartimento di Elettronica, Politecnico di Milano, 1991.

[9] P.Rosenstiehl and R.E.Tarjan, Rectilinear planar layouts of planar graphs and bipolar orientations, *Discr. Comp. Geometry* **1**, 343–353, 1986.

Received November 8, 1992

Acta Cybernetica, Vol. 10, Nr. 4, Szeged, 1992

On the Boolean structure of fuzzy logical systems: a counter example

J. Dombi *† Gy. Lencsés ‡

Abstract

The article of Murthy, Pal and Majumder [1] gives a new interpretation of the connectives in fuzzy sets claiming that these connectives preserve the whole Boolean structure of ordinary set theoretic operations. In our paper a counter example is given where the property of associativity is not valid for the new connectives.

Introduction

Many authors attempt to construct fuzzy logical systems preserving as many Boolean properties as it is possible. It is well known that to preserve the whole Boolean structure of set operations when extending them pointwisely to [0,1] valued membership functions of fuzzy sets is not possible (see e.g. [2]). For instance excluded middle law and idempotence are incompatible for fuzzy sets if we demand that the result of the operation in any point must be dependent only on the value of the membership functions in this point.

C.A. Murthy et al. [1] try to solve this problem by defining operators the result of which may be dependent not only on the value of membership functions but also on their relative natures. They claim that the operators $\textcircled{\cap}$ and $\textcircled{\cup}$ defined in their article fulfil all of the Boolean properties. If it were true, then these new operators should be preferred to any other earlier construction.

We will show, however, a counter example where the operators $\textcircled{\cap}$ and $\textcircled{\cup}$ do not fulfil some Boolean properties. It will be shown that the operators are ill-defined, and we will point out why it is impossible to prove some of the Boolean properties of the operators $\textcircled{\cap}$ and $\textcircled{\cup}$ by the Theorems 1-7 of the cited paper.

*Research Group on the Theory of Automata, Hungarian Academy of Sciences Aradi Vértanúk tere 1., H-6720 Szeged, Hungary

†This work was carried out during my stay in FRG and supported by the Humboldt Foundation.

‡Department of Sociology, University of Szeged, Petőfi S. sgt. 30-34, H-6720 Szeged, Hungary

1 Preliminary definitions

First of all we have to recall the definitions of C.A. Murthy et al. [1].

1. 1 Properties of the operators

They claim that the operators $\textcircled{\cap}$ and $\textcircled{\cup}$ fulfil the following properties. Here A, B, C are fuzzy sets in a universe X, μ_A, μ_B, etc. are membership functions of A, B, etc., A^C is the complement of A.

p_1 : $\mu_A \ \textcircled{\cap} \ _{A^C}(x) = 0$ for all $x \in X$

p_2 : $\mu_A \ \textcircled{\cup} \ _{A^C}(x) = 1$ for all $x \in X$

p_3 : commutativity

$\mu_A \ \textcircled{\cup} \ _B(x) = \mu_B \ \textcircled{\cup} \ _A(x)$

$\mu_A \ \textcircled{\cap} \ _B(x) = \mu_B \ \textcircled{\cap} \ _A(x)$

p_4 : associativity

$\mu_A \ \textcircled{\cup} \ (B \ \textcircled{\cup} \ C)(x) = \mu_{(A \ \textcircled{\cup} \ B)} \ \textcircled{\cup} \ _C(x)$

$\mu_A \ \textcircled{\cap} \ (B \ \textcircled{\cap} \ C)(x) = \mu_{(A \ \textcircled{\cap} \ B)} \ \textcircled{\cap} \ _C(x)$

p_5 : idempotency

$\mu_A \ \textcircled{\cup} \ _A(x) = \mu_A(x)$

$\mu_A \ \textcircled{\cap} \ _A(x) = \mu_A(x)$

p_6 : distributive laws

$\mu_A \ \textcircled{\cup} \ (B \ \textcircled{\cap} \ C)(x) = \mu_{(A \ \textcircled{\cup} \ B)} \ \textcircled{\cap} \ (A \ \textcircled{\cup} \ C)(x)$

$\mu_A \ \textcircled{\cap} \ (B \ \textcircled{\cup} \ C)(x) = \mu_{(A \ \textcircled{\cap} \ B)} \ \textcircled{\cup} \ (A \ \textcircled{\cap} \ C)(x)$

p_7 : identity

$\mu_A \ \textcircled{\cup} \ \emptyset(x) = \mu_A(x)$

$\mu_A \ \textcircled{\cap} \ _X(x) = \mu_A(x)$

p_8 : a) absorption laws

b) DeMorgan's laws

c) involution laws

p_9 : $0 \leq \mu_A \ \textcircled{\cap} \ _B \leq \min(\mu_A, \mu_B)$

$1 \geq \mu_A \ \textcircled{\cup} \ _B \geq \max(\mu_A, \mu_B)$

In the following μ_A, μ_B, μ_C are denoted by f,g,h respectively, $\mu_A \ \textcircled{\cap} \ _B$ is denoted by $f \ \textcircled{\cap} \ g$, etc.

1. 2 Definition of type I membership functions

Let the domain $Q = [a, b]$ be a closed interval in R, and let f be a membership function with the following properties:

a) $f : Q \to [0,1]$ is continuous
b) $f(Q) = [0,1]$
c) $f\{a,b\} \subseteq \{0,1\}$

f is a type I membership function if it fulfils the next assumption:
Let $a < x_0 < b$ such that f increases (decreases) at x_0. Then there exist x_1 and x_2 such that

$$a \leq x_1 < x_0 < x_2 \leq b \quad \text{and} \quad f(x_1) = 0 \ (f(x_1) = 1), \ f(x_2) = 1$$

$(f(x_2) = 0)$ and f is nondecreasing (nonincreasing) at all $x \in (x_1, x_2)$.

1. 3 Definition of $\textcircled{\cap}$ and $\textcircled{\cup}$.

a) Murthy et al. first define a set A_x for every f membership function and for every point $x \in [a,b]$ as follows:

$$A_x = \begin{cases} [0, f(x)] & \text{if } f \text{ is nondecreasing at } x \\ [1 - f(x), 1] & \text{if } f \text{ is nonincreasing at } x \\ \text{any finite set} & \text{if } f(x) = 0 \\ [0,1] & \text{if } f(x) = 1 \end{cases}$$

B_x and C_x are similarly defined for the functions g and h in any point x.

b) Then $\textcircled{\cap}$ and $\textcircled{\cup}$ are defined by

$$(f \ \textcircled{\cap} \ g)(x) = \lambda(A_x \cap B_x)$$
$$(f \ \textcircled{\cup} \ g)(x) = \lambda(A_x \cup B_x)$$

where λ is the Lebesgue measure on R.

2 A counter example

We give an example, where the property of associativity (P_4) of $\textcircled{\cap}$ does not hold.
Let us consider the $\textcircled{\cap}$ operator. If we use it two times, one after another

$$((f \ \textcircled{\cap} \ g) \ \textcircled{\cap} \ h)(x)$$

then according to the definition 1.3 in each step first of all we have to determine sets:

– in the first step: the sets A_x and B_x,
– in the second step: the set D_x connected with $(f \ \textcircled{\cap} \ g)(x)$ by definition 1.3a, and the set C_x.

But it is easily possible that $D_x = A_x \cap B_x$ is not valid (e.g. when f is increasing and g is decreasing at x). In this case the properties of Lebesgue measure in connection with ordinary sets cannot be automatically used to prove associativity and distributivity as it was done in Part VI of the cited paper.
Let us see a counter example where f, g and h are type I membership functions and the associativity of $\textcircled{\cap}$ does not hold. Let $Q = [0,1]$, f, g and h be piecewise linear membership functions as shown in Figure 1.

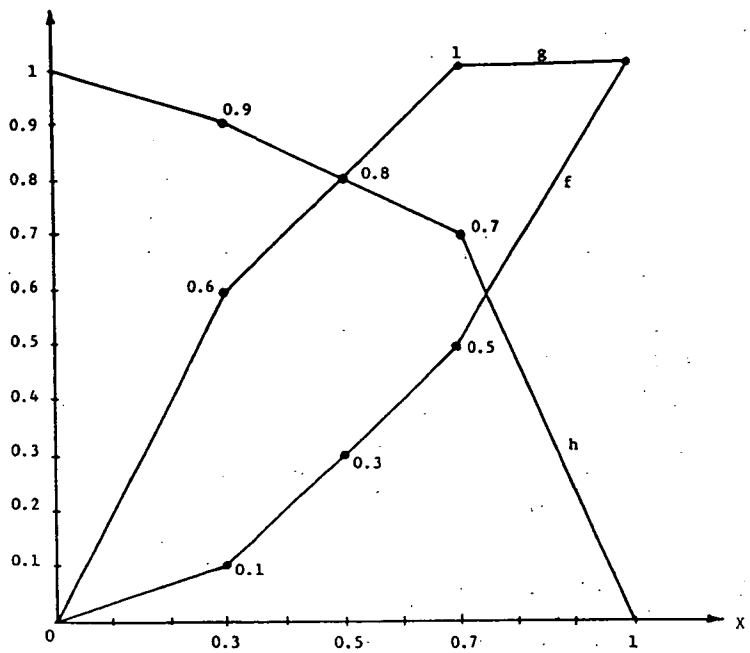

Fig. 1. Functions f, g, h

Here f, g and h are type I membership functions. But $g \textcircled{\cap} h$ does not belong to the same type because $(g \textcircled{\cap} h)([0,1]) = [0, 0.7]$. See figure 2.

Since for all $x \in [0,1]$ $f(x) \leq g(x)$, and f and g are nondecreasing at all $x \in (0,1)$, so $f \textcircled{\cap} g = f$ on the whole interval $[0,1]$, according to the definition of $\textcircled{\cap}$. On the interval $(0.3, 0.7)$ the functions $f, g, f \textcircled{\cap} g (= f), g \textcircled{\cap} h$ are nondecreasing and h is nonincreasing.

So $(f \textcircled{\cap} g)(0.5) = f(0.5) = 0.3$.

Let the set connected to $f \textcircled{\cap} g$ at the point 0.5 be $D_{0.5}$ (see definition 1.3a). Then $D_{0.5} = [0, 0.3]$, because $f \textcircled{\cap} g$ is nondecreasing at 0.5. $C_{0.5} = [0.2, 1]$, because h is nonincreasing at 0.5.

So $((f \textcircled{\cap} g) \textcircled{\cap} h)(0.5) = \lambda(D_{0.5} \cap C_{0.5}) = 0.1$

Similarly since $g \textcircled{\cap} h$ is increasing at 0.5, and $(g \textcircled{\cap} h)(0.5) = 0.6$, so $E_{0.5} = [0, 0.6]$, where $E_{0.5}$ is the set connected to $g \textcircled{\cap} h(0.5)$ by definition 1.3a. In addition $f(0.5) = 0.3$, $A_{0.5} = [0, 0.3]$ and so

$$(f \textcircled{\cap} (g \textcircled{\cap} h))(0.5) = \lambda(A_{0.5} \cap E_{0.5}) = 0.3$$

That is $((f \textcircled{\cap} g) \textcircled{\cap} h)(0.5) \neq (f \textcircled{\cap} (g \textcircled{\cap} h)(0.5)$

This result contradicts to the property of associativity of $\textcircled{\cap}$.

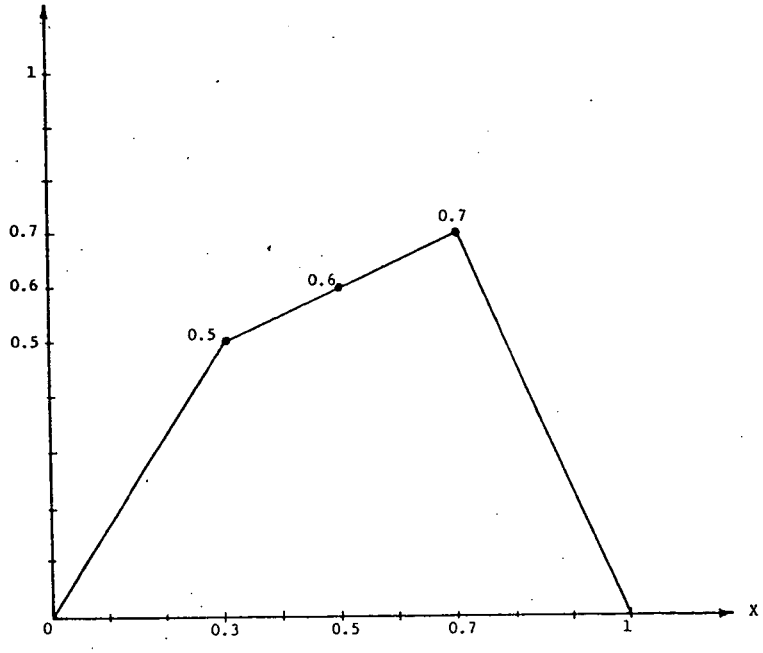

Fig. 2. Function $g \textcircled{\cap} h$

3 Concluding remarks

1. The definition of $\textcircled{\cap}$ and $\textcircled{\cup}$ is suitable only for type I membership functions. How can the set E_x be determined for the function $g \textcircled{\cap} h$ in Fig. 2. at the point $x = 0.7$? Here $g \textcircled{\cap} h$ attains its maximum value, $g \textcircled{\cap} h(0.7) = 0.7$, but this value is not equal to 1.

2. Why is the proof of associativity wrong in [1]? The cited paper uses the following argumentation to prove the Boolean properties of $\textcircled{\cap}$ and $\textcircled{\cup}$: "the operations are ordinary set operations and the Lebesgue measure satisfies similar properties in connection with ordinary sets". This reasoning would be correct only if in composite operations the sets A_x, B_x, etc. connected to the membership functions were inherited. That is if $f(x) = (g \textcircled{\cap} h)(x)$ and A_x, B_x, C_x are obtained from definition 1.3a, then $A_x = B_x \cap C_x$ for all $x \in [a, b]$. Murthy et al. prove this only for the case when f, g and h are type I membership functions (see Theorems 1-7 in [1]). If, however, the result function f does not belong to the same type then the above equality for the sets A_x, B_x, C_x is not true usually. (See e.g. the function $g \textcircled{\cap} h$ in our counter example.)

References

[1] C. A. Murthy, S. K. Pal and D. D. Majumder, "Representation of fuzzy operators using ordinary sets", IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-17 no. 5, pp. 840-847, 1987.

[2] D.Dubois and H. Prade, "New results about properties and semantics of fuzzy set-theoretic operations", in P.P. Wang and S.K. Chang, Eds., Fuzzy Sets, Pergamon Press, 1980.

Received March 19, 1993

Acta Cybernetica, Vol. 10, Nr. 4, Szeged, 1992

On minimal and maximal clones

L. Szabó*†

1 Introduction

A composition closed set of finitary operations on a fixed universe A containing all projections is a clone. For example the set $\mathbf{J}$ of all projections and the set $\mathbf{O}$ of all operations on A are clones. The clones, ordered by inclusion, form an algebraic lattice $\mathbf{L}$ with least element $\mathbf{J}$ and greatest element $\mathbf{O}$. For $|A| = 2$, $\mathbf{L}$ is the well-known countable Post lattice [5], but already for $|A| > 2$ there are $2^{\aleph_0}$ clones. For A finite $\mathbf{L}$ has finitely many coatoms, called maximal clones , and they are fully known ([7],[8]). On the other hand $\mathbf{L}$ has finitely many atoms, called minimal clones, and are fully known only for $|A| \leq 3$ ([3], [5]). It is also known (see e.g. [6]) that the meet of all maximal clones is $\mathbf{J}$, and the join of all minimal clones is $\mathbf{O}$.

The aim of the present paper is to show that in general there are three maximal clones with meet $\mathbf{J}$ and there are three minimal clones with join $\mathbf{O}$; moreover, for a prime element universe, two maximal clones, resp., two minimal clones have the above properties.

2 Preliminaries

Let A be a fixed universe with $|A| \geq 2$. For any positive integer n let $\mathbf{O}^{(n)}$ denote the set of all n-ary operations on A (i.e. maps $A^n \to A$) and let $\mathbf{O} = \bigcup_{n=1}^{\infty} \mathbf{O}^{(n)}$. For $1 \leq i \leq n$ let e_i^n denote the n-ary i-th projection (trivial operation). Further let $\mathbf{J} = \{e_i^n | 1 \leq i \leq n < \infty\}$. The operations in $\mathbf{O} \setminus \mathbf{J}$ are called *nontrivial operations*. By a *clone* we mean a subset of $\mathbf{O}$ which is closed under superpositions and contains all projections. The set of clones ordered by inclusion form a lattice $\mathbf{L}$ in which every meet is the set-theoretical intersection. For $F \subseteq \mathbf{O}$ denote by $[F]$ the clone generated by F, and instead of $[\{f\}]$ we write $[f]$.

A minimal clone, resp., a maximal clone is an atom, resp., a dual atom of $\mathbf{L}$. It is well-known that $\mathbf{L}$ is an atomic and dually atomic algebraic lattice, and has finitely many minimal clones and maximal clones. Furthermore, the intersection of all maximal clones is $\mathbf{J}$, and the minimal clones generate $\mathbf{O}$ (see e.g. [6]). The maximal clones are fully known and was given by I. G. Rosenberg ([7], [8]). For $|A| = 2$, $\mathbf{L}$ is the well-known Post lattice [5]. Considering the Post lattice we immediately see that for two element set there are three maximal clones with intersection $\mathbf{J}$ and the intersection of two maximal clones cannot be $\mathbf{J}$. Moreover, there are three minimal clones with join $\mathbf{O}$ and the join of two minimal clones cannot be $\mathbf{O}$.

*Bolyai Institute, Aradi vértanúk tere 1, 6720 Szeged, Hungary

†Research partially supported by Hungarian National Foundation for Scientific Research Grant no. 1813 and 1903.

A subset $F \subseteq \mathbf{O}$ as well as the algebra (A, F) is *primal* or *complete* if the clone generated by F (i.e. the set of all term functions of (A, F)) is equal to $\mathbf{O}$; F as well as the algebra (A, F) is *functionally complete* if F together with all constant operations is primal.

A ternary operation f on A is a *majority function* if for all $x, y \in A$ we have $f(x, x, y) = f(x, y, x) = f(y, x, x) = x$; f is a *Mal'tsev function* if $f(x, y, y) = f(y, y, x) = x$ for all $x, y \in A$. An n-ary operation t on A is said to be an i-th *semi-projection* $(n \geq 3,\ 1 \leq i \leq n)$ if for all $x_1, \ldots, x_n \in A$ we have $t(x_1, \ldots, x_n) = x_i$ whenever at least two elements among $x_1, \ldots, x_n$ are equal. We are going to formulate Rosenberg's Theorem ([7], [8]) which is the main tool in proving our results. First, however, we need some further definitions.

Let $n, h \geq 1$. An n-ary operation $f \in \mathbf{O}^{(n)}$ is said to *preserve* the h-ary relation $\rho \subseteq A^h$ if ρ is a subalgebra of the h-th direct power of the algebra $(A; f)$. Then the set of operations preserving ρ forms a clone, which is denoted by $\mathrm{Pol}\rho$. We say that a relation ρ is a *compatible relation* of the algebra (A, F) if $F \subseteq \mathrm{Pol}\rho$. A binary relation is called *nontrivial* if it is distinct from the identity relation and from the full relation.

An h-ary relation ρ on A is called *central* if $\rho \neq A^h$ and there exists a non-void proper subset C of A (called the *center* of ρ) such that

(a) $(a_1, \ldots, a_h) \in \rho$ whenever at least one $a_i \in C (1 \leq i \leq h)$;

(b) ρ is *totally symmetric*, i.e. $(a_1, \ldots, a_h) \in \rho$ implies $(a_{1\pi}, \ldots, a_{h\pi}) \in \rho$ for every permutation ϕ of the indices $1, \ldots, h$;

(c) ρ is *totally reflexive*, i.e. $(a_1, \ldots, a_h) \in \rho$ if $a_i = a_j$ for some $i \neq j (1 \leq i, j \leq h)$.

Let $h \geq 3$. A family $T = \{\Theta_1, \ldots, \Theta_m\}$ $(m \geq 1)$ of equivalence relations on A is called h-*regular* if each Θ_i $(1 \leq i \leq m)$ has exactly h blocks and $\Theta_T = \Theta_1 \cap \ldots \cap \Theta_m$ has exactly h^m blocks (i.e. the intersection $\bigcap_{i=1}^m B_i$ of arbitrary blocks B_i of $\Theta_i (i = 1, \ldots, m)$ is nonempty). The relation determined by T is

$$\lambda_T = \{(a_1, \ldots, a_h) \in A^h : a_1, \ldots, a_h \text{ are not pairwise incongruent}$$

$$\text{modulo } \Theta_i \text{ for all } i (1 \leq i \leq m)\}.$$

Note that h-regular relations are both totally reflexive and totally symmetric.

Now we are in a position to state Rosenberg's Theorem:

Theorem A (I. G. Rosenberg [7],[8]). *A subset of* $\mathbf{O}$ *is a maximal clone if and only if it is of the form* $\mathrm{Pol}\rho$ *for a relaton* ρ *of one of the following six types:*

1. a *bounded partial order;*
2. a *binary relation* $\{(a, a\pi) | a \in A\}$ *where* π *is a permutation of* A *with* $|A|/p$ *cycles of the same length* p *(*p *is a prime number);*
3. a *quaternary relation* $\{(a_1, a_2, a_3, a_4) \in A^4 | a_1 + a_2 = a_3 + a_4\}$ *where* $(A; +)$ *is an elementary abelian* p*-group (*p *is a prime number);*
4. a *nontrivial equivalence relation;*
5. a *central relation;*
6. a *relation determined by an* h*-regular family of equivalence relations.*

Moreover, a finite algebra $\mathbf{A} = (A, F)$ *is primal if and only if* $F \subseteq \mathrm{Pol}\rho$ *for no relation* ρ *of any of the above six types.*

3 Results

From now on A is supposed to be the set $\{0, \ldots, k-1\}$ with $k > 2$.

Theorem 3.1 *There exist three maximal clones such that their intersection is* **J**. *Moreover, if* k *is a prime number then there are two maximal clones such that their intersection is* **J**.

Proof. For any $a \in A$ define a binary relation ρ_a on A as follows:

$$\rho_a = \{(x, y) | x = a \text{ or } y = a \text{ or } x = y\}.$$

Observe that ρ_a is a central relation with center $\{a\}$. Choose two fixed point free permutation σ and τ on A of prime orders such that $\{\sigma, \tau\}$ generates a transitive permutation group on A. If k is a prime number then we can choose σ and τ with $\sigma = \tau$. Then, by Theorem A, $\mathrm{Pol}\rho_a$ $(a \in A)$, $\mathrm{Pol}\sigma$ and $\mathrm{Pol}\tau$ are maximal clones. Put $F = \mathrm{Pol}\rho_0 \cap \mathrm{Pol}\sigma \cap \mathrm{Pol}\tau$. We show that $F = \mathbf{J}$.

Consider the algebra $\mathbf{A} = (A; F)$. Then ρ_0 is a compatible relation, σ and τ are automorphisms of $\mathbf{A}$. Therefore, by the choice of σ and τ, Aut $\mathbf{A}$ is transitive, which implies that every operation of $\mathbf{A}$ is surjective. Moreover, if $\pi \in \mathrm{Aut}\ \mathbf{A}$ then

$$\rho_{0\pi} = \{(x\pi, y\pi) | (x, y) \in \rho_0\}$$

is also a compatible relation of $\mathbf{A}$. Therefore, by the transitivity of Aut $\mathbf{A}$, we have that ρ_a is a compatible relation of $\mathbf{A}$ for every $a \in A$. From this it follows that for every distinct $a, b \in A$

$$\rho_{ab} = \rho_a \cap \rho_b = \{(a, b), (b, a)\} \cup \{(x, x) | x \in A\}$$

is also a compatible relation of $\mathbf{A}$.

It is well-known that if a surjective operation preserves a central relation then it preserves its center (see e.g. [9]). Thus we have that every operation in F is idempotent. Suppose that $\mathbf{A}$ has a nontrivial operation. Then it has either a nontivial binary operation or a majority function or a Mal'tsev function or a nontrivial semi-projection among its term functions (see e.g. [4]).

First consider the case when $\mathbf{A}$ has a nontrivial binary term function f. Let $a, b \in A$ be arbitrary distinct elements. Then from $(a, b), (b, b) \in \rho_{ab}$ we have that $(f(a, b), b) = (f(a, b), f(b, b)) \in \rho_{ab}$, implying that $f(a, b) = a$ or $f(a, b) = b$. Suppose that $f(a, b) = a$ and choose an arbitrary element $c \in A$ with $c \neq a, b$. Then $(a, a), (b, c) \in \rho_{bc}$ implies that $(f(a, c), a) = (f(a, c), f(a, b)) \in \rho_{bc}$ and $f(a, c) = a$. This fact together with the transitivity of Aut $\mathbf{A}$ shows that f is the first projection, a contradiction. If $f(a, b) = b$ then a similar argument yields that f is the second projection.

Now let d be a majority term function of $\mathbf{A}$, and let $a, b, c \in A$ be pairwise different elements. Then $d(a, b, c)$ is different from two of the elements a, b, c, say from a and b. Then $(a, a), (b, a), (c, c) \in \rho_b$ implies that $(d(a, b, c), a) = (d(a, b, c), d(a, a, c)) \in \rho_b$, a contradiction.

If t is a Mal'tsev function among the term functions of $\mathbf{A}$, then for any two distinct elements $a, b \neq 0$, $(a, 0), (0, 0), (0, b) \in \rho_0$ implies that $(a, b) = (t(a, 0, 0), (t(0, 0, b)) \in \rho_0$, a contradiction.

Finally, let l be a nontrivial n-ary first semi-projection among the term functions of $\mathbf{A}$. Since l is not the first projection, there are $a, a_2, \ldots, a_n \in A$ such that $l(a, a_2, \ldots, a_n) = b \neq a$. Choose $c \in A$ with $c \neq a, b$. Then $(a, c), (a_2, a), \ldots, (a_n, a) \in \rho_a$ implies that $(b, c) = (l(a, a_2, \ldots, a_n), l(c, a, \ldots, a)) \in \rho_a$, a contradiction. This completes the proof.

Theorem 3.2 *There exist three minimal clones such that their join is* $\mathbf{O}$. *Moreover, if* k *is a prime number then there are two minimal clones such that their join is* $\mathbf{O}$.

Proof. First consider the case when k is a prime number and let σ be the permutation $(0\ 1\ \ldots\ k-1)$ on A. Clearly, $[\sigma]$ is a minimal clone. Define a ternary operation f on A as follows:

$$f(x, y, z) = \max(\min(x, y), \min(x, z), \min(y, z))$$

for all $x, y, z \in A$. Then f is a majority function and $[f]$ is a minimal clone (see e.g. [6]). We show that f together with σ generates $\mathbf{O}$. Put $F = \{f, \sigma\}$.

Taking into consideration Theorem A, we have to show that $F \subseteq \mathrm{Pol}\rho$ for no relation of any of the types (1)-(6). Since σ generates a transitive permutation group, it is easy to show that it cannot preserve a relation of type (1) and (5). Moreover, making use of the fact that k is a prime number, one can show easily that σ do not preserve a relation of type (4). Furthermore, f being a majority function - as it is well-known (see e.g. [4]) - do not preserve a relation of type (3) and (6). Finally suppose that ρ is a relation of type (2) determined by a permutation π with $F \subseteq \mathrm{Pol}\rho$. Then π is an automorphism of the algebra $\mathbf{A} = (A; F)$. Since π and σ commute we have that π is a power of σ, and then σ is also a power of π (k is prime). Hence σ is an automorphism of $\mathbf{A}$. Therefore, we have that $1 = f(0, 1, 2) = f((k-1)\sigma, 0\sigma, 1\sigma) = f(k-1, 0, 1)\sigma = 1\sigma = 2$, a contradiction. This completes the proof when k is a prime number.

Now suppose that k is not a prime, and let p be a prime number such that $k/2 < p < k$. Consider the permutations $\sigma = (0\ 1\ \ldots\ p-1)$ and $\tau = (k-p\ \ k-(p-1)\ \ldots\ k-1)$ on A. Clearly, $[\sigma]$ and $[\tau]$ are minimal clones. Define a ternary operation d on A as follows:

$$d(x, y, z) = \begin{cases} x, & \text{if } x = y, \\ z, & \text{otherwise.} \end{cases}$$

Then d is the well-known dual discriminator, which generates a minimal clone (see e.g. [2]). We show that σ and τ together with d generate $\mathbf{O}$. Put $F = \{d, \sigma, \tau\}$.

Again, by Theorem A, we have to show that $F \subseteq \mathrm{Pol}\rho$ for no relation of any of the types (1)-(6). Suppose that $F \subseteq \mathrm{Pol}\rho$ for a relation of one of the type (1)-(6). It is known that $\{d\}$ is a functionally complete set (see e. g. [1]). Therefore $d \notin \mathrm{Pol}\rho$ if $\mathrm{Pol}\rho$ contains all constant operations. Hence ρ is of type (2) or a unary central relation. Since $[\sigma]$ and $[\tau]$ generate a transitive permutation group, they do not preserve a unary central relation.

Finally suppose that ρ is a relation of type (2) determined by a permutation π. Observe that if π is of order q then π is the product of k/q cycles of the same length q. Moreover, since k is not a prime number, we have $q \leq k/2$. Then π commutes with σ and τ. Let $0\pi = i$. If $i > p-1$ then for all $j \in \{0, 1, \ldots, p-1\}$ we have $j\pi = 0\sigma^j\pi = 0\pi\sigma^j = i\sigma^j = i$, showing that π is not injective, a contradiction. Hence $i < p$, and for all $j \in \{0, 1, \ldots, p-1\}$ we have $j\pi = 0\sigma^j\pi = 0\pi\sigma^j = i\sigma^j = 0\sigma^i\sigma^j = 0\sigma^j\sigma^i = j\sigma^i$ showing that π contains the cycle σ^i of length p. Therefore we have $p = q \leq k/2$, a contradiction. This completes the proof.

Problem 1 *Find all natural numbers k for which there exist two maximal clones on the set $\{0,\dots,k-1\}$ such that their intersection is* $\mathbf{J}$.

Problem 2 *Find all natural numbers k for which there exist two minimal clones on the set $\{0,\dots,k-1\}$ such that their join is* $\mathbf{O}$.

References

[1] B. Csákány, Homogeneous algebras are functionally complete, *Algebra Universalis* **11** (1980), 149-158.

[2] B. Csákány and T. Gavalcová, Finite homogeneous algebras. I, *Acta Sci. Math.* **42** (1980), 57-65.

[3] B. Csákány, All minimal clones on the three-element set, *Acta Cybernet.* **6** (1980), 227-238.

[4] P. P. Pálfy, L. Szabó and Á. Szendrei, Automorphism groups and functonal completeness, *Algebra Universalis* **15** (1982), 385-400.

[5] E. L. Post, *The two-valued iterative systems of mathematical logic*, Ann. Math. Studies 5, Princeton Univ. Press, 1941.

[6] R. Pöschel and L. A. Kalouznin, *Functionen- und Relationenalgebren*, VEB Deutscher Verlag d. Wissenschaften, Berlin, 1979.

[7] I. G. Rosenberg, Über die funktionale Vollständigkeit in den mehrwertigen Logiken (Struktur der Funktionen von mehreren Veränderlichen auf endlichen Mengen), *Rozpravy Československe Akad. Věd Řada Mat. Přírod. Věd* **80** (1970), 9–93.

[8] I. G. Rosenberg, Completeness properties of multiple-valued logic algebras, in: *Computer Science and Multiple-Valued Logic, Theory and Applications* (ed. D. C. Rine), North-Holland (1977); pp. 144-186.

[9] I. G. Rosenberg, Functional completeness of single generated or surjective algebras, in: *Finite Algebra and Multiple-Valued Logic* (Proc. Conf. Szeged, 1979), Colloq. Math. Soc. J. Bolyai, vol. 28, North-Holland, Amsterdam, 1981; pp. 635–652.

Received September 26, 1991.

Készítette a JATEPress
6722 Szeged, Petőfi Sándor sugárút 30—34.

Subscription information and mailing address for editorial correspondence:

Acta Cybernetica
Árpád tér 2.
Szeged
H-6720 Hungary

CONTENTS

A. Ádám: A criterion for the simplicity of finite Moore automata 221
B. Imreh: On a special composition of tree automata 237
J. Dassow, A. Mateescu, G. Paun, A. Salomaa: Regularizing context-free languages by AFL operations: concatenation and Kleene closure 243
E. Jurvanen: The Boolean Closure of DR-Recognizable Tree Languages 255
B. Almási: A Queuing Model for a Processor-Shared Multi-Terminal System Subject to Breakdowns 273
T. Makjamroen: The Self-organizing List and Processor Problems under Randomized Policies 283
G. J. Woeginger: Computing Maximum Valued Regions 303
J. Dombi, Gy. Lencsés: On the Boolean structure of fuzzy logical systems: a counter example 317
L. Szabó: On minimal and maximal clones 323

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1993. július
Terjedelem: 7,12 (B/5) ív